Guia do Desenvolvedor

Banco de dados Amazon Quantum Ledger (AmazonQLDB)

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Banco de dados Amazon Quantum Ledger (AmazonQLDB): Guia do Desenvolvedor

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é o Amazon QLDB? ... 1
QLDBVídeo da Amazon ... 1
QLDBPreços da Amazon ... 2
Conceitos básicos do QLDB ... 2
Visão geral ... 2
Diário primeiro ... 3
Imutável ... 4
Verificável criptograficamente ... 5
SQL-semelhante e flexível em documentos ... 6
Ferramentas de desenvolvedor de código aberto ... 7
Tecnologia sem servidor e altamente disponível ... 7
Nível empresarial ... 7
Do relacional ao ledger ... 7
Conceitos principais ... 10
QLDBmodelo de objeto de dados ... 11
Transações que priorizam o diário ... 13
Consultar seus dados ... 14
Armazenamento de dados ... 14
QLDBAPImodelo ... 15
Próximas etapas ... 16
Conteúdo do diário ... 16
Exemplo do bloco ... 17
Conteúdo do bloco ... 19
Revisões editadas ... 21
Aplicação de exemplo ... 23
Consulte também ... 23
Glossário da QLDB ... 24
Acessando a Amazon QLDB ... 28
Pré-requisitos ... 28
Inscreva-se para um Conta da AWS ... 28
Criar um usuário com acesso administrativo ... 29
Gerenciar QLDB permissões em IAM ... 30
Conceder acesso programático ... 30
Como acessar a Amazon QLDB ... 32

Usar o console ... 33
Referência rápida do editor PartiQL ... 33
Usar o AWS CLI (APIsomente gerenciamento) ... 38
Instalando e configurando o AWS CLI ... 39
Usar o AWS CLI com QLDB ... 39
Usando o Amazon QLDB shell (APIsomente dados) ... 40
Pré-requisitos ... 41
Instalar o shell ... 42
Invocar o shell ... 42
Parâmetros do shell ... 43
Referência de comando ... 44
Executar instruções individuais ... 46
Gerenciamento de transações ... 46
Sair do shell ... 48
Exemplo ... 48
Usando o API ... 49
Como iniciar com o console do ... 50
Pré-requisitos e considerações ... 51
Configurar permissões ... 52
Etapa 1: criar um novo ledger ... 53
Etapa 2: criar tabelas, índices e dados de amostra ... 55
Etapa 3: consultar as tabelas ... 64
Etapa 4: Modificar documentos ... 66
Etapa 5: Visualizar o histórico de revisão ... 69
Etapa 6: verificar um documento ... 72
Para solicitar um resumo ... 73
Para verificar a revisão de um documento ... 74
Etapa 7: limpar ... 76
Próximas etapas ... 77
Conceitos básicos do driver ... 78
driver Java ... 79
Recursos para driver ... 80
Pré-requisitos ... 80
Como credenciar suas credenciais AWS e região padrão ... 81
Instalação ... 82
Tutorial de início rápido ... 85

Referência de Cookbook ... 93
. NETmotorista ... 110
Recursos para driver ... 111
Pré-requisitos ... 111
Instalação ... 112
Tutorial de início rápido ... 113
Referência de Cookbook ... 138
Driver Go ... 171
Recursos para driver ... 172
Pré-requisitos ... 172
Instalação ... 173
Tutorial de início rápido ... 174
Referência de Cookbook ... 186
driver Node.js ... 201
Recursos para driver ... 201
Pré-requisitos ... 201
Instalação ... 202
Recomendações de configuração ... 207
Tutorial de início rápido ... 210
Referência de Cookbook ... 229
Driver Python ... 251
Recursos para driver ... 252
Pré-requisitos ... 252
Instalação ... 253
Tutorial de início rápido ... 255
Referência de Cookbook ... 261
Gerenciamento da sessão com o driver ... 274
Ciclo de vida da sessão ... 275
Expiração da sessão ... 276
Tratamento da sessão no QLDB driver ... 276
Recomendações de driver ... 279
Configurando o objeto QldbDriver ... 279
Tentar novamente com exceções ... 282
Otimizar o desempenho ... 283
Como executar várias instruções por transação ... 284
Política de novas tentativas de driver ... 288

Tipos de erros que podem ser tentados novamente ... 289
Política padrão de tentativas ... 289
Erros comuns ... 290
Tutorial de aplicativo de exemplo ... 293
Tutorial de Java ... 294
Tutorial Node.js ... 464
Tutorial do Python ... 527
Como trabalhar com o Amazon Ion ... 614
Pré-requisitos ... 615
Bool ... 616
Int ... 619
Float ... 622
Decimal ... 627
Timestamp ... 630
String ... 634
Blob ... 638
Listar ... 641
struct ... 645
Valores nulos e tipos dinâmicos ... 652
Conversão descendente para JSON ... 657
Trabalhar com dados e histórico ... 658
Criação de tabelas com índices e inserção de documentos ... 659
Criar tabelas e índices ... 660
Inserir documentos ... 661
Consultar seus dados ... 663
Consultas básicas ... 664
Projeções e filtros ... 666
Junções ... 667
Dados aninhados ... 668
Consultar metadados do documento ... 669
Visão confirmada ... 670
Unir as visualizações confirmadas e do usuário ... 672
Usar a cláusula BY para consultar o ID do documento ... 673
Ingressando no ID do documento ... 674
Atualizar e excluir documentos ... 674
Fazer revisões de documentos ... 675

Consultar o histórico de revisões 676
Função de histórico 677
Exemplo de consulta de histórico 678
Redigir revisões de documentos 681
Procedimentos de redação armazenados 682
Verificar se uma redação está completa 683
Exemplo de redação 683
Excluir e editar uma revisão ativa 686
Editar um campo específico em uma revisão 686
Otimizar a performance da consulta 687
Tempo limite de transação 688
Conflitos de concorrência 688
Padrões de consulta ideais 688
Padrões de consulta a serem evitados 690
Monitorar o desempenho 691
Obter estatísticas de instruções PartiQL 691
Uso de E/S 692
Informações de cronometragem 699
Consultar o catálogo do sistema 705
Gerenciar tabelas 707
Marcar tabelas na criação 707
Eliminar tabelas 708
Consultando o histórico de tabelas inativas 708
Reativando tabelas 709
Gerenciamento de Índices 709
Criar índices 710
Descrever índices 711
Reduzir índices 713
Erros comuns 714
Único IDs 715
Propriedades 715
Uso 715
Exemplos 716
Modelo de simultaneidade 717
Controle de simultaneidade otimista 717
Usar índices para evitar varreduras completas da tabela 718

Conflitos de inserção OCC ... 719
Tornar as transações idempotentes ... 721
Conflitos de redação OCC ... 721
Gerenciar sessões simultâneas ... 722
Verificação ... 723
Em que tipo de dados você pode verificarQLDB? ... 724
O que significa integridade de dados? ... 725
Como a verificação funciona? ... 725
Hashing ... 725
Resumo ... 727
Árvore Merkle ... 727
Prova ... 728
Exemplo de verificação ... 728
Como a redação de dados afeta a verificação? ... 729
Como recalcular um hash de revisão ... 730
Introdução à verificação ... 730
Etapa 1: Solicitar um resumo ... 731
AWS Management Console ... 731
QLDB API ... 733
Etapa 2: Verificar seus dados ... 734
AWS Management Console ... 734
QLDB API ... 736
Resultados da verificação ... 737
Como usar uma prova para recalcular seu resumo ... 739
Tutorial: Verificando dados usando um AWS SDK ... 739
Pré-requisitos ... 740
Etapa 1: Solicite um resumo ... 741
Etapa 2: Consultar a revisão do documento ... 742
Etapa 3: Solicitar uma prova da revisão ... 744
Etapa 4: Recalcular o resumo da revisão ... 749
Etapa 5: Solicitar uma prova para o bloco de diário ... 751
Etapa 6: Recalcular o resumo do bloco ... 756
Execute o exemplo de código completo ... 764
Erros comuns ... 788
Exportação de dados do diário ... 791
Solicitação de uma exportação ... 792

AWS Management Console ... 792
QLDB API ... 795
Expiração do trabalho de exportação ... 797
Saída da exportação ... 797
Arquivos de manifesto ... 798
Objetos de dados ... 800
Conversão descendente para JSON ... 804
Biblioteca de processadores de exportação (Java) ... 804
Permissões de exportação ... 804
Criação de uma política de permissões ... 806
Crie uma IAM função ... 808
Erros comuns ... 810
Fluxos ... 813
Casos de uso comuns ... 814
Consumindo seu fluxo ... 815
Garantia de entrega ... 815
Considerações sobre latência de entrega ... 815
Introdução aos fluxos ... 816
Criar e gerenciar fluxos ... 816
Parâmetros de fluxo ... 817
Transmitir ARN ... 819
AWS Management Console ... 819
Estados de fluxo ... 821
Tratamento de fluxos prejudicados ... 822
Desenvolver com fluxos ... 823
QLDBfluxo de diário APIs ... 824
Aplicações de exemplo ... 825
Registros de fluxo ... 827
Registros de controle ... 828
Bloquear registros resumidos ... 829
Registros de detalhes da revisão ... 831
Lidando com duplicatas e registros out-of-order ... 832
Permissões de fluxo ... 833
Criação de uma política de permissões ... 834
Crie uma IAM função ... 836
Erros comuns ... 839

Gerenciamento de ledgers ... 842
Operações básicas para ledgers ... 842
Criar um ledger ... 843
Descrever um ledger ... 847
Atualizar um ledger ... 850
Atualizar o modo de permissões de um ledger ... 853
Excluir um ledger ... 855
Listar ledgers ... 856
AWS CloudFormation recursos ... 857
QLDBe AWS CloudFormation modelos ... 858
Saiba mais sobre AWS CloudFormation ... 858
Marcar recursos ... 859
Recursos suportados na Amazon QLDB ... 860
Convenções de uso e nomenclatura de tags ... 860
Como gerenciar tags ... 861
Atribuição de tags a recursos durante a criação ... 861
Segurança ... 863
Proteção de dados ... 864
Criptografia em repouso ... 865
Criptografia em trânsito ... 883
Identity and Access Management ... 884
Público ... 885
Autenticando com identidades ... 885
Gerenciando acesso usando políticas ... 889
Como a Amazon QLDB trabalha com IAM ... 892
Introdução a do modo de permissões padrão ... 901
Exemplos de políticas baseadas em identidade ... 913
Prevenção do problema do substituto confuso entre serviços ... 932
AWS políticas gerenciadas ... 935
Solução de problemas ... 939
Logging e monitoramento ... 941
Ferramentas de monitoramento ... 943
Monitoramento com a Amazon CloudWatch ... 945
Automatização com eventos CloudWatch ... 950
Registrando QLDB API chamadas da Amazon com AWS CloudTrail ... 951
Validação de conformidade ... 971

Resiliência ... 973
Durabilidade do armazenamento ... 973
Atributos de durabilidade de dados ... 973
Segurança da infraestrutura ... 975
AWS PrivateLink ... 975
Solução de problemas ... 980
Executando transações usando o QLDB driver ... 980
Exportação de dados do diário ... 983
Streaming de dados do diário ... 986
Verificação dos dados do diário ... 987
Referência do PartiQL ... 990
O que é PartiQL? ... 991
PartiQL na Amazon QLDB ... 991
Dicas rápidas do partiQL em QLDB ... 991
Convenções Referência do PartiQL ... 992
Tipos de dados ... 993
QLDBdocumentos ... 995
Estrutura de documento Ion ... 995
Mapeamento do tipo de PartiQL-Ion ... 996
ID do documento ... 997
Consultar o Ion com o PartiQL ... 997
Sintaxe e semântica ... 998
Notação de acento grave ... 1000
Navegação por caminhos ... 1002
Aliases ... 1002
Especificação PartiQL ... 1003
Comandos PartiQL ... 1003
DDLdeclarações ... 1004
DMLdeclarações ... 1004
CREATE INDEX ... 1004
CREATE TABLE ... 1007
DELETE ... 1010
DROP INDEX ... 1012
DROP TABLE ... 1013
FROM(INSERT,REMOVE, ouSET) ... 1015
INSERT ... 1020

SELECT ... 1024
UPDATE ... 1030
UNDROP TABLE ... 1035
Funções PartiQL ... 1036
Funções agregadas ... 1037
Funções condicionais ... 1037
Perfis de data e hora ... 1037
Funções escalares ... 1037
Funções de string ... 1037
Funções de formatação de tipo de dados ... 1038
AVG ... 1038
CAST ... 1039
CHAR_LENGTH ... 1043
CHARACTER_LENGTH ... 1044
COALESCE ... 1045
COUNT ... 1046
DATE_ADD ... 1047
DATE_DIFF ... 1049
EXISTS ... 1051
EXTRACT ... 1052
LOWER ... 1054
MAX ... 1055
MIN ... 1056
NULLIF ... 1058
SIZE ... 1059
SUBSTRING ... 1061
SUM ... 1062
PARA_ STRING ... 1064
PARA_ TIMESTAMP ... 1065
TRIM ... 1067
TXID ... 1069
UPPER ... 1070
UTCNOW ... 1071
Strings de formato da data e hora ... 1072
Procedimentos armazenados PartiQL ... 1074
REDACT_REVISION ... 1075

operadores PartiQL 1079
Operadores aritméticos 1079
Operadores de comparação 1079
Operadores lógicos 1080
Operadores de string 1081
Palavras-chave reservadas 1081
Referência do Amazon Ion 1087
O que é o Amazon Ion? 1088
Especificação de Ion 1089
JSONcompatível 1089
Extensões de JSON 1089
Exemplo de texto Ion 1091
APIreferências 1091
Exemplos de código do Amazon Ion 1092
APIreferência 1107
Ações 1108
Amazon QLDB 1109
QLDBSessão Amazon 1184
Tipos de dados 1192
Amazon QLDB 1193
QLDBSessão Amazon 1210
Erros comuns 1233
Parâmetros gerais 1234
Cotas e limites 1238
Cotas padrão 1238
Cotas fixas 1239
Cota do ledger 1240
Tamanho do documento 1241
Tamanho da transação 1241
Restrições de nomenclatura 1242
Informações relacionadas 1243
Documentação técnica 1243
GitHub repositórios 1244
AWS postagens e artigos do blog 1245
Mídia 1247
AWS Recursos gerais 1249

Histórico de versões 1250
..... mcclxxi

O que é o Amazon QLDB?

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O Amazon Quantum Ledger Database (AmazonQLDB) é um banco de dados contábil totalmente gerenciado que fornece um registro de transações transparente, imutável e criptograficamente verificável, de propriedade de uma autoridade central confiável. Você pode usar QLDB a Amazon para rastrear todas as alterações nos dados do aplicativo e manter um histórico completo e verificável das alterações ao longo do tempo. Para saber mais sobre a variedade de opções de bancos de dados disponíveis na Amazon Web Services, consulte Como escolher o banco de dados certo para sua organização na AWS.

Os ledgers são normalmente usados para registrar um histórico da atividade econômica e financeira em uma organização. Muitas organizações criam aplicativos com funcionalidade semelhante a um ledger porque desejam manter um histórico preciso dos dados de seus aplicativos. Por exemplo, elas podem querer rastrear o histórico de créditos e débitos em transações bancárias, verificar a linhagem de dados de uma reclamação de seguro ou rastrear a movimentação de um item em uma rede da cadeia de suprimentos. Os aplicativos de ledger geralmente são implementados usando tabelas de auditoria personalizadas ou trilhas de auditoria criadas em bancos de dados relacionais.

QLDBA Amazon é uma nova classe de banco de dados que ajuda a eliminar a necessidade de se engajar no complexo esforço de desenvolvimento de criar seus próprios aplicativos semelhantes a livros contábeis. ComQLDB, o histórico de alterações em seus dados é imutável — não pode ser sobrescrito ou alterado no local. E, usando criptografia, você pode verificar se não houve alterações não intencionais nos dados do seu aplicativo. QLDBusa um registro transacional imutável, conhecido como diário. O diário é somente para anexar e é composto por um conjunto de blocos sequenciados e encadeados em hash que contêm seus dados confirmados.

QLDBVídeo da Amazon

Para uma visão geral da Amazon QLDB e como ela pode beneficiar você, assista a este vídeo de QLDB visão geral em YouTube.

QLDBPreços da Amazon

Com a AmazonQLDB, você paga somente pelo que usa, sem taxas mínimas ou uso obrigatório do serviço. Você paga apenas pelo armazenamento e E/S consumidos pelo banco de dados ledger, sem necessidade de provisionamento antecipado.

Para obter mais informações, consulte os QLDBpreços da Amazon.

Conceitos básicos do QLDB

Recomendamos que você comece lendo os seguintes tópicos:

- Visão geral da Amazon QLDB— Para obter uma visão geral de alto nível de. QLDB
- Conceitos e terminologia fundamentais na Amazon QLDB— Aprender QLDB conceitos e terminologia fundamentais.
- Acessando a Amazon QLDB— Para saber como acessar QLDB usando o AWS Management Console,API, ou AWS Command Line Interface (AWS CLI).
- Como a Amazon QLDB trabalha com IAM— Para aprender a controlar o acesso ao QLDB uso de AWS Identity and Access Management (IAM).

Para começar rapidamente a usar o QLDB console, consulteComeçando a usar o QLDB console da Amazon.

Para saber mais sobre o desenvolvimento QLDB usando um driver AWS fornecido, consulteComeçando com o QLDB driver da Amazon.

Visão geral da Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

As seções a seguir fornecem uma visão geral de alto nível dos componentes de QLDB serviços da Amazon e de como eles interagem.

Tópicos

- Diário primeiro
- Imutável
- Verificável criptograficamente
- SQL-semelhante e flexível em documentos
- Ferramentas de desenvolvedor de código aberto
- Tecnologia sem servidor e altamente disponível
- Nível empresarial

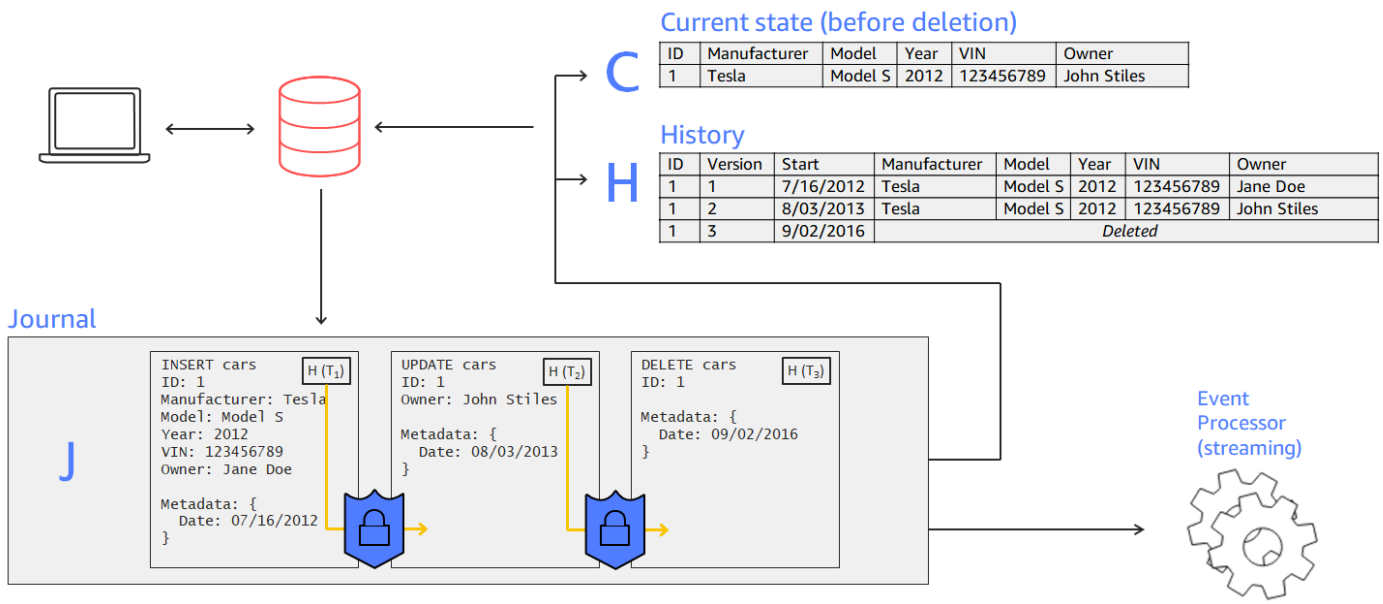
Diário primeiro

Na arquitetura tradicional de banco de dados, você geralmente grava dados em tabelas como parte de uma transação. Um log de transações — normalmente uma implementação interna — registra todas as transações e as modificações do banco de dados que elas fazem. O log de transações é um componente essencial do banco de dados. Você precisa do log para reproduzir transações em caso de falha do sistema, recuperação de desastres ou replicação de dados. No entanto, os logs de transações do banco de dados não são imutáveis e não foram projetados para fornecer acesso direto e fácil aos usuários.

Na AmazonQLDB, a revista é o núcleo do banco de dados. Estruturalmente semelhante a um log de transações, o diário é uma estrutura de dados imutável, somente para anexar, que armazena os dados do seu aplicativo junto com os metadados associados. Todas as transações de gravação, incluindo atualizações e exclusões, são confirmadas primeiro no diário.

QLDBusa o diário para determinar o estado atual dos dados contábeis, materializando-os em tabelas consultáveis e definidas pelo usuário. Essas tabelas também fornecem um histórico acessível de todos os dados da transação, incluindo revisões de documentos e metadados. Além disso, o diário lida com simultaneidade, sequenciamento, verificação criptográfica e disponibilidade dos dados do ledger.

O diagrama a seguir ilustra a arquitetura do QLDB periódico.



- Neste exemplo, um aplicativo se conecta a um ledger e executa transações que inserem, atualizam e excluem um documento em uma tabela chamada `cars`.
- Os dados são primeiro gravados no diário em ordem sequenciada.
- Em seguida, os dados são materializados na tabela com visualizações integradas. Essas visualizações permitem que você consulte o estado atual e o histórico completo do carro, comum número de versão atribuído a cada revisão.
- Você também pode exportar ou transmitir dados diretamente do diário.

Imutável

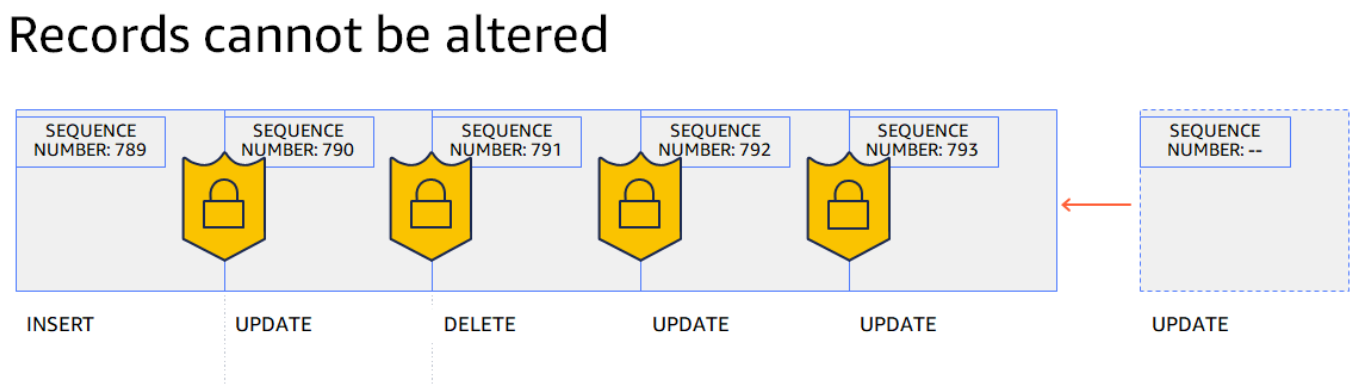
Como o QLDB diário é somente para anexar, ele mantém um registro completo de todas as alterações em seus dados que não podem ser modificadas ou sobrescritas. Não existem APIs ou outros métodos para alterar os dados confirmados. Essa estrutura de diário permite acessar e consultar o histórico completo do seu ledger.

> **Note**
>
> A única exceção à imutabilidade QLDB compatível é a redação de dados. Com esse recurso, você pode cumprir os estatutos regulatórios, como o Regulamento Geral de Proteção de

Dados (GDPR) na União Europeia e a Lei de Privacidade do Consumidor da Califórnia (CCPA).
QLDBfornece uma operação de redação que permite excluir permanentemente revisões de documentos inativas no histórico de uma tabela. Essa operação exclui somente os dados do usuário na revisão especificada e deixa a sequência do diário e os metadados do documento inalterados. Isso mantém a integridade geral dos dados do seu ledger. Para ter mais informações, consulte Redigir revisões de documentos.

QLDBgrava um bloco no diário em uma transação. Cada bloco contém objetos de entrada que representam os documentos que você insere, atualiza e exclui, junto com as instruções que você executou para confirmá-las. Esses blocos são sequenciados e encadeados em hash para garantir a integridade dos dados.

O diagrama a seguir ilustra essa estrutura do diário.

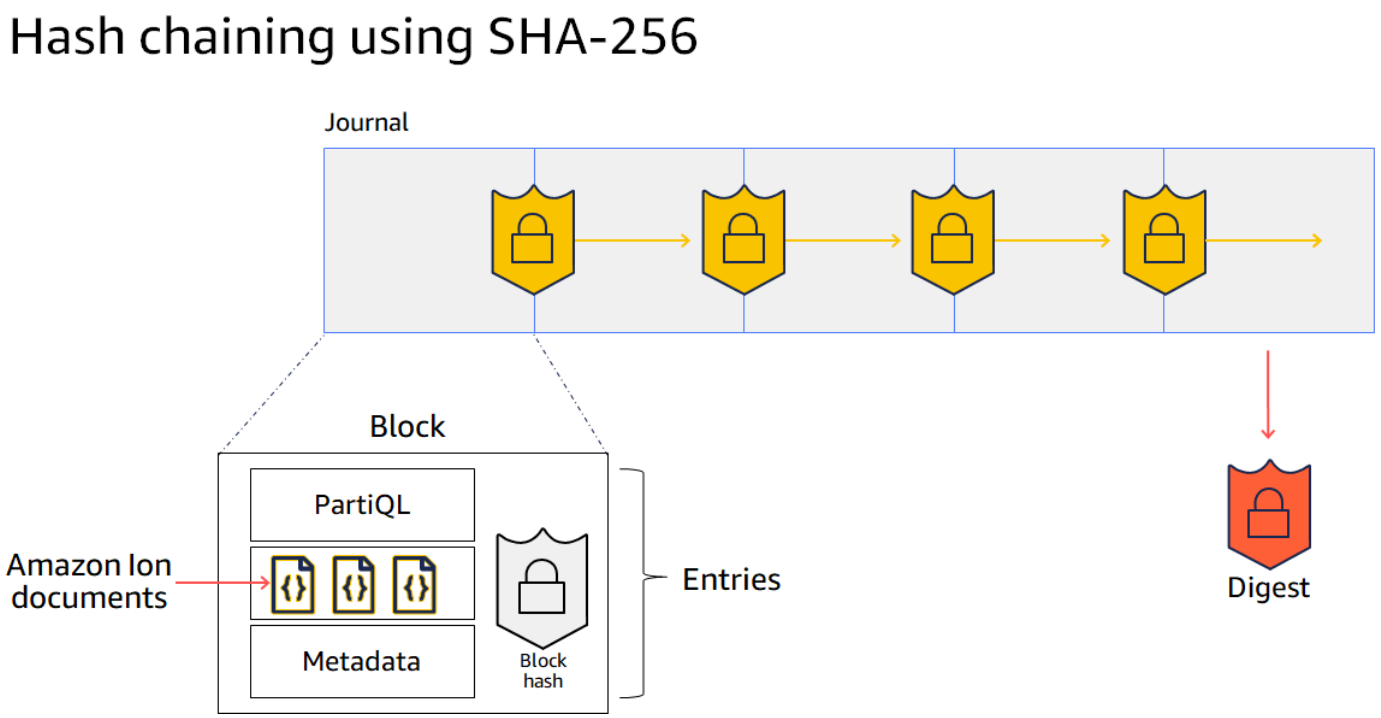


O diagrama mostra que as transações são confirmadas no diário como blocos que são encadeados em hash para verificação. Cada bloco tem um número de sequência para especificar seu endereço.

Verificável criptograficamente

Os blocos de diário são sequenciados e encadeados com técnicas criptográficas de hashing, semelhantes às blockchains. QLDBusa a cadeia de hash do diário para fornecer integridade de dados transacionais usando um método de verificação criptográfica. Usando um resumo (um valor de hash que representa a cadeia de hash completa de um diário em um determinado momento) e uma prova de auditoria Merkle (um mecanismo que comprova a validade de qualquer nó em uma árvore de hash binária), você pode verificar se não houve alterações não intencionais em seus dados em qualquer momento.

O diagrama a seguir mostra um resumo que abrange toda a cadeia de hash de um diário em um determinado momento.



Neste diagrama, os blocos de diário são codificados usando a função hash criptográfica SHA -256 e são sequencialmente encadeados aos blocos subsequentes. Cada bloco contém entradas que incluem seus documentos de dados, metadados e as instruções partiQL executadas na transação.

Para ter mais informações, consulte Verificação de dados na Amazon QLDB.

SQL-semelhante e flexível em documentos

QLDBusa o PartiQL como sua linguagem de consulta e o Amazon Ion como seu modelo de dados orientado a documentos. O partiQL é uma linguagem de consulta de código aberto SQL compatível que foi estendida para funcionar com o Ion. Com o partiQL, você pode inserir, consultar e gerenciar seus dados com operadores conhecidosSQL. Quando você está consultando documentos simples, a sintaxe é a mesma usada SQL para consultar tabelas relacionais. Para saber mais sobre a QLDB implementação do partiQL, consulte o. Referência do Amazon QLDB PartiQL

Amazon Ion é um superconjunto de. JSON O Ion é um formato de dados de código aberto baseado em documentos que oferece a flexibilidade de armazenar e processar dados estruturados, semiestruturados e aninhados. Para saber mais sobre o Ion inQLDB, consulte Referência de formato de dados Amazon Ion na Amazon QLDB o.

Para uma comparação de alto nível dos principais componentes e recursos dos bancos de dados relacionais tradicionais versusQLDB, consulte. Do relacional ao ledger

Ferramentas de desenvolvedor de código aberto

Para simplificar o desenvolvimento de aplicativos, QLDB fornece drivers de código aberto em várias linguagens de programação. Você pode usar esses drivers para interagir com os dados API transacionais executando instruções partiQL em um livro contábil e processando os resultados dessas instruções. Para obter informações e tutoriais sobre as linguagens de driver atualmente suportadas, consulte Começando com o QLDB driver da Amazon.

O Amazon Ion também fornece bibliotecas de clientes que processam os dados do Ion para você. Para guias de desenvolvedores e exemplos de código de processamento de dados do Ion, consulte a documentação do Amazon Ion em GitHub.

Tecnologia sem servidor e altamente disponível

QLDBé totalmente gerenciado, sem servidor e altamente disponível. O serviço é escalado automaticamente para atender às demandas do seu aplicativo, e você não precisa provisionar instâncias ou capacidade. Várias cópias de seus dados são replicadas dentro de uma zona de disponibilidade e entre zonas de disponibilidade em uma Região da AWS.

Nível empresarial

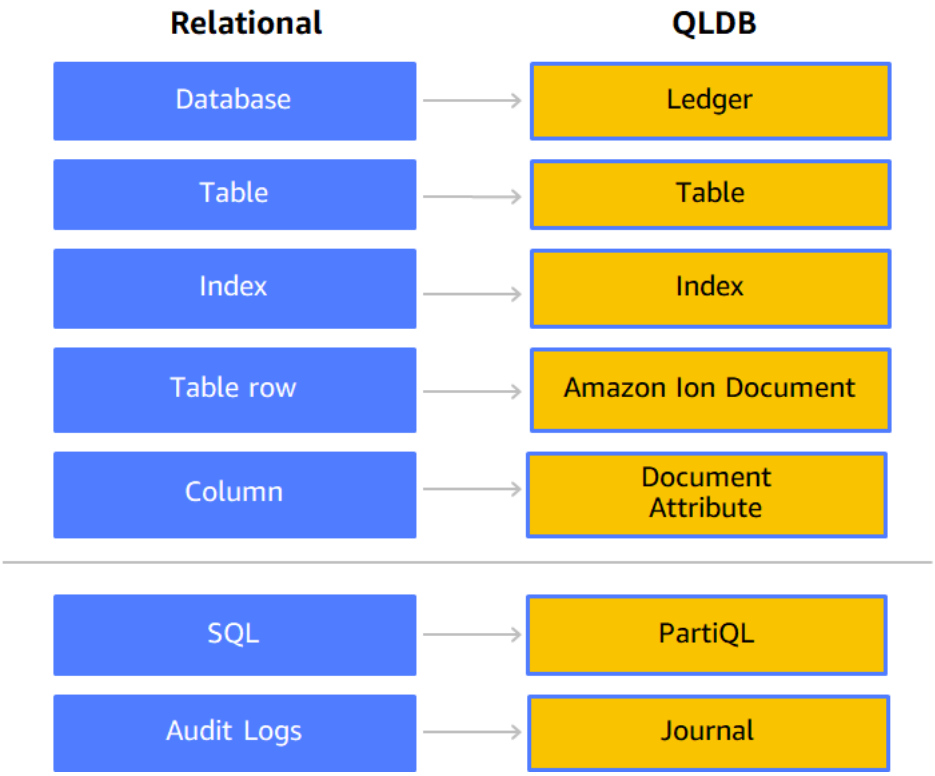
QLDBas transações são totalmente compatíveis com as propriedades de atomicidade, consistência, isolamento e durabilidade (ACID). QLDBusa controle de simultaneidade otimista (OCC), e as transações operam com serialização total — o mais alto nível de isolamento. Isso significa que não há risco de ver leituras fantasmas, leituras sujas, distorção de gravação ou outros problemas de simultaneidade semelhantes. Para ter mais informações, consulte Modelo de QLDB concorrência da Amazon.

Do relacional ao ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Se você é desenvolvedor de aplicativos, talvez tenha alguma experiência no uso de um sistema de gerenciamento de banco de dados relacional (RDBMS) e da Structured Query Language (SQL). Ao começar a trabalhar com a AmazonQLDB, você encontrará muitas semelhanças. À medida que avança para tópicos mais avançados, você também encontrará novos recursos poderosos QLDB que foram construídos sobre a RDBMS base. Esta seção descreve os componentes e operações comuns do banco de dados, comparando-os e contrastando-os com seus equivalentes em. QLDB

O diagrama a seguir mostra as construções de mapeamento dos componentes principais entre um sistema tradicional RDBMS e a AmazonQLDB.



A tabela a seguir mostra as principais semelhanças e diferenças de alto nível dos recursos operacionais integrados entre um tradicional e. RDBMS QLDB

Operation	RDBMS	QLDB
Criar tabelas	Instrução `CREATE TABLE` que define todos os nomes de colunas e os tipos de dados	Instrução `CREATE TABLE` que não define nenhum atributo de tabela ou tipo de dados para permitir conteúdo aberto e sem esquemas

Operation	RDBMS	QLDB
Criar índices	Instrução `CREATE INDEX`	Instrução `CREATE INDEX` para qualquer campo de nível superior em uma tabela
Inserir dados	Instrução `INSERT` que especifica valores em uma nova linha ou tupla que adere ao esquema conforme definido pela tabela	Instrução `INSERT` que especifica valores em um novo documento em qualquer formato válido do Amazon Ion, independentemente dos documentos existentes na tabela
Consultar dados	Instrução `SELECT-FROM-WHERE`	`SELECT-FROM-WHERE`declaração na mesma sintaxe da consulta de documentos simples SQL
Atualização de dados	Instrução `UPDATE-SET-WHERE`	`UPDATE-SET-WHERE`declaração na mesma sintaxe da atualização de documentos simples SQL
Excluir dados	Instrução `DELETE-FROM-WHERE`	`DELETE-FROM-WHERE`declaração na mesma sintaxe de SQL ao excluir documentos simples
Dados aninhados e semiestruturados	Somente linhas planas ou tuplas	Documentos que podem ter qualquer dado estruturado, semiestruturado ou aninhado, conforme suportado pelo formato de dados Amazon Ion e pela linguagem de consulta PartiQL

Operation	RDBMS	QLDB
Consultar metadados	Não há metadados integrados	Instrução `SELECT` que consulta a partir da visualiza ção comprometida integrada de uma tabela
Consultar o histórico de revisões	Não há histórico de dados integrado	Instrução `SELECT` que consulta a partir da função de histórico integrada
Verificação criptográfica	Sem criptografia ou imutabili dade incorporadas	APIsque retornam um resumo de um diário e uma prova que verifica a integridade de qualquer revisão do documento em relação a esse resumo

Para obter uma visão geral dos principais conceitos e terminologia emQLDB, consulteConceitos principais.

Para obter informações detalhadas sobre o processo de criação, consulta e gerenciamento de seus dados em um ledger, consulte Trabalhar com dados e histórico.

Conceitos e terminologia fundamentais na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção fornece uma visão geral dos principais conceitos e terminologia da AmazonQLDB, incluindo a estrutura do livro contábil e como um livro contábil gerencia os dados. Como banco de dados contábil, QLDB difere de outros bancos de dados orientados a documentos quando se trata dos seguintes conceitos-chave.

Tópicos

- QLDBmodelo de objeto de dados
- Transações que priorizam o diário
- Consultar seus dados
- Armazenamento de dados
- QLDBAPImodelo
- Próximas etapas

QLDBmodelo de objeto de dados

O modelo de objeto de dados fundamental na Amazon QLDB é descrito a seguir:

1. ledger

   Sua primeira etapa é criar um livro contábil, que é o principal tipo de AWS recurso emQLDB. Para saber como criar um ledger, consulte Etapa 1: criar um novo ledger em Conceitos básicos do console, ou Operações básicas para livros contábeis da Amazon QLDB.

   Para os modos de registro `ALLOW_ALL` e `STANDARD` permissões de um livro contábil, você cria políticas AWS Identity and Access Management (IAM) que concedem permissões para executar API operações nesse recurso contábil.

   ARNFormato do livro contábil:

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

2. Diário e tabelas

   Para começar a gravar dados em um QLDB livro contábil, primeiro você cria uma tabela com uma CREATE TABLE declaração básica. Os dados do ledger consistem em revisões de documentos que estão comprometidos com o diário do ledger. Você confirma as revisões do documento no ledger no contexto de tabelas definidas pelo usuário. EmQLDB, uma tabela representa uma visão materializada de uma coleção de revisões de documentos do diário.

   No modo de `STANDARD` permissões de um livro contábil, você deve criar IAM políticas que concedam permissões para executar instruções partiQL nesse recurso de tabela. Com permissões em um recurso de tabela, você pode executar instruções que acessam o estado atual da tabela.

Você também pode consultar o histórico de revisões da tabela usando a função `history()` incorporada.

ARNFormato da tabela:

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

Para obter mais informações sobre como conceder permissões em um ledger e seus recursos associados, consulte Como a Amazon QLDB trabalha com IAM.

3. Documentos

   As tabelas consistem em QLDBdocumentos, que são conjuntos de dados no formato Amazon Ion `struct`. Uma revisão de documento representa uma única versão de uma sequência de documentos identificados por um ID de documento exclusivo.

   QLDBarmazena o histórico completo de alterações de seus documentos confirmados. Uma tabela permite consultar o estado atual de seus documentos, enquanto a função `history()` permite consultar todo o histórico de revisão dos documentos de uma tabela. Para obter detalhes sobre como consultar e escrever revisões, consulte Trabalhar com dados e histórico.

4. Catálogo do sistema

   Cada ledger também fornece um recurso de catálogo definido pelo sistema que você pode consultar para listar todas as tabelas e índices em um ledger. No modo de permissões `STANDARD` de um ledger, você precisa da permissão `qldb:PartiQLSelect` desse recurso de catálogo para fazer o seguinte:

   - Executar instruções `SELECT` na tabela do catálogo do sistema information_schema.user_tables.
   - Veja as informações da tabela e do índice na página de detalhes do livro contábil no QLDBconsole.
   - Veja a lista de tabelas e índices no editor partiQL no QLDB console.

   ARNFormato do catálogo:

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

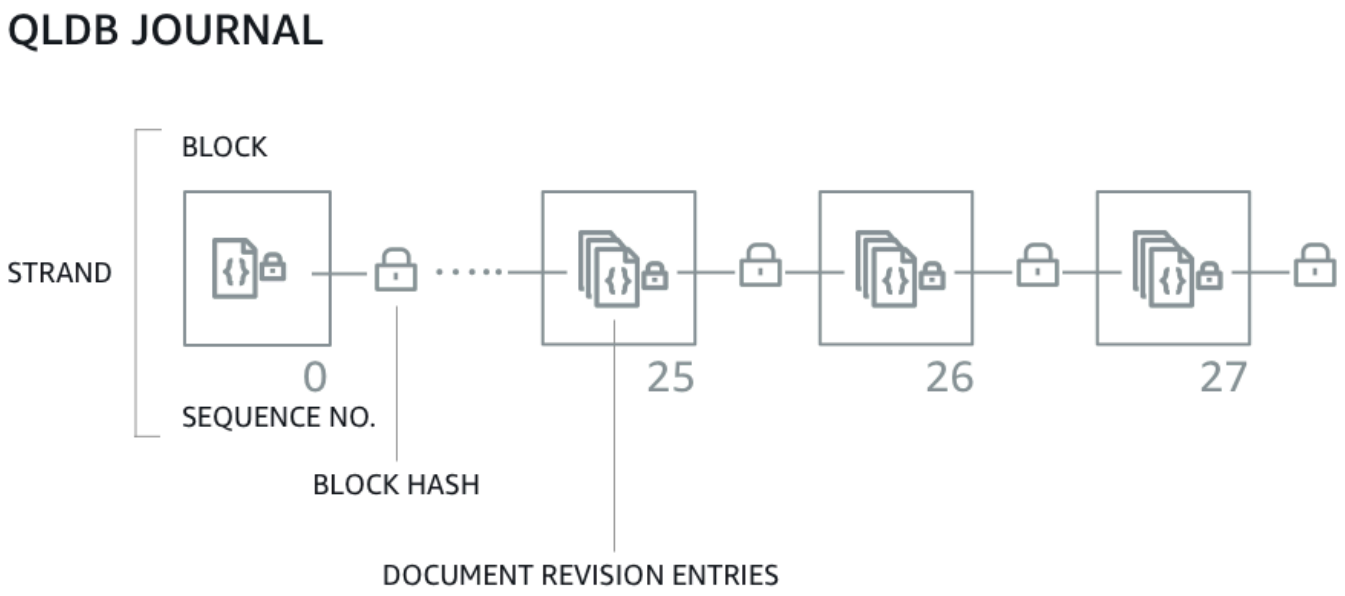
Transações que priorizam o diário

Quando um aplicativo lê ou grava dados em um QLDB livro contábil, ele o faz em uma transação de banco de dados. Todas as transações estão sujeitas aos limites definidos em Cotas e limites na Amazon QLDB. Em uma transação, QLDB execute as seguintes etapas:

1. Lê o estado atual dos dados do ledger.
2. Execute as instruções fornecidas na transação e, em seguida, verifique se há conflitos usando o controle otimista de simultaneidade (OCC) para garantir um isolamento totalmente serializável.
3. Se nenhum OCC conflito for encontrado, retorne os resultados da transação da seguinte forma:
   - Para leituras, retorne o conjunto de resultados e confirme as instruções `SELECT` no diário apenas como anexo.
   - Para gravações, confirme quaisquer atualizações, exclusões ou dados recém-inseridos no diário de forma somente para anexar.

O diário representa um histórico completo e imutável de todas as alterações aos seus dados. QLDBgrava um bloco encadeado no diário em uma transação. Cada bloco contém objetos de entrada que representam as revisões do documento que você insere, atualiza e exclui, junto com as instruções partiQL que as confirmaram.

O diagrama a seguir ilustra essa estrutura do diário.



O diagrama mostra que as transações são confirmadas no diário como blocos que contêm entradas de revisão do documento. Cada bloco é criptografado e encadeado aos blocos subsequentes para verificação. Cada bloco tem um número de sequência para especificar seu endereço dentro da cadeia.

> **Note**
>
> Na AmazonQLDB, um fio é uma partição do diário do seu livro contábil. QLDBatualmente suporta periódicos com apenas uma única vertente.

Para obter mais informações sobre o conteúdo de dados em um bloco, consulte Conteúdo do periódico na Amazon QLDB.

Consultar seus dados

QLDBtem como objetivo atender às necessidades de cargas de trabalho de processamento de transações on-line (OLTP) de alto desempenho. Um ledger fornece visualizações de tabela consultáveis de seus dados com base nas informações da transação que estão confirmadas no diário. Uma exibição de tabela em QLDB é um subconjunto dos dados em uma tabela. As visualizações são mantidas em tempo real, para que estejam sempre disponíveis para consulta dos aplicativos.

Você pode consultar as seguintes visualizações definidas pelo sistema usando instruções `SELECT` PartiQL:

- Usuário: a revisão ativa mais recente somente dos dados que você gravou na tabela (ou seja, o estado atual dos dados do usuário). Essa é a visualização padrão emQLDB.
- Confirmada: a revisão ativa mais recente dos dados do usuário e dos metadados gerados pelo sistema. Essa é a tabela completa definida pelo sistema que corresponde diretamente à sua tabela de usuário.

Além dessas visualizações que podem ser consultadas, você pode consultar o histórico de revisões de seus dados usando o Função de histórico integrado. A função de histórico retorna os dados do usuário e os metadados associados no mesmo esquema da visualização confirmada.

Armazenamento de dados

Há dois tipos de armazenamento de dados emQLDB:

- Armazenamento de diário: o espaço em disco usado pelo diário de um ledger. O diário é somente para anexar e representa um histórico completo, imutável e verificável de todas as alterações aos seus dados.
- Armazenamento indexado: o espaço em disco usado pelas tabelas, índices e histórico indexado de um ledger. O armazenamento indexado consiste em dados do ledger otimizados para consultas de alta performance.

Depois que seus dados são confirmados no diário, eles são materializados nas tabelas que você definiu. Essas tabelas são otimizadas para consultas mais rápidas e eficientes. Quando um aplicativo usa os dados transacionais API para ler dados, ele acessa as tabelas e os índices armazenados no seu armazenamento indexado.

QLDBAPImodelo

QLDBfornece dois tipos com APIs os quais seu código de aplicativo pode interagir:

- Amazon QLDB — O gerenciamento de QLDB recursos API (também conhecido como plano de controle). Isso API é usado somente para gerenciar recursos contábeis e para operações de dados não transacionais. Você pode usar essas operações para criar, excluir, descrever, listar e atualizar ledgers. Você também pode verificar dados do diário criptograficamente e exportar ou transmitir blocos de diário.
- Amazon QLDB Session — Os QLDB dados transacionais. API Você pode usar isso API para executar transações de dados em um livro contábil com instruções partiQL.

> **Important**
>
> Em vez de interagir diretamente com a QLDBsessãoAPI, recomendamos usar o QLDB driver ou o QLDB shell para executar transações de dados em um livro contábil.
>
> - Se você estiver trabalhando com um AWS SDK, use o QLDB driver. O driver fornece uma camada de abstração de alto nível acima dos dados da QLDBsessão API e gerencia a `SendCommand` operação para você. Para obter informações e uma lista das linguagens de programação suportadas, consulte Conceitos básicos do driver.
> - Se você estiver trabalhando com o AWS CLI, use o QLDB shell. O shell é uma interface de linha de comando que usa o QLDB driver para interagir com um livro contábil. Para obter mais informações, consulte Usando o Amazon QLDB shell (APIsomente dados).

Para obter mais informações sobre essas API operações, consulte QLDBAPIReferência da Amazon o.

Próximas etapas

Para aprender a usar um ledger com seus dados, veja Trabalhando com dados e histórico na Amazon QLDB e siga os exemplos que descrevem o processo de criar tabelas, inserir dados e executar consultas básicas. Este guia explica detalhadamente como esses conceitos funcionam, usando dados de exemplo e exemplos de consulta para contexto.

Para começar rapidamente com um exemplo de tutorial de aplicativo usando o QLDB console, consulteComeçando a usar o QLDB console da Amazon.

Para obter uma lista dos principais termos e definições descritos nesta seção, consulte o QLDBGlossário da Amazon.

Conteúdo do periódico na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, o diário é o registro transacional imutável que armazena o histórico completo e verificável de todas as alterações em seus dados. O diário é somente para anexar e é composto por um conjunto de blocos sequenciados e encadeados em hash que contêm seus dados confirmados e outros metadados do sistema. QLDBgrava um bloco encadeado no diário em uma transação.

Esta seção fornece um exemplo de um bloco de diário com dados de amostra e descreve o conteúdo de um bloco.

Tópicos

- Exemplo do bloco
- Conteúdo do bloco
- Revisões editadas
- Aplicação de exemplo

- Consulte também

Exemplo do bloco

Um bloco de diário contém metadados da transação junto com entradas que representam as revisões do documento que foram confirmadas na transação e as instruções partiQL que as confirmaram.

Veja a seguir um exemplo de um bloco com dados de amostra.

> **Note**
>
> Este exemplo de bloco é fornecido apenas para fins informativos. Os hashes mostrados não são valores reais de hash calculados.

```
{
  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"
      }
    }
  ]
}
```

No campo `revisions`, alguns objetos de revisão podem conter apenas um valor `hash` e nenhum outro atributo. Essas são revisões de sistema somente internas que não contêm dados do usuário. Os hashes dessas revisões fazem parte da cadeia de hash completa do diário, necessária para a verificação criptográfica.

Conteúdo do bloco

Um bloco de diário tem os seguintes campos:

`blockAddress`

A localização do bloco no diário. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

`transactionId`

O ID exclusivo da transação que confirmou o bloco.

`blockTimestamp`

A data e hora em que o bloco foi confirmado no diário.

`blockHash`

O valor de hash de 256 bits que representa exclusivamente o bloco. Esse é o hash da concatenação de `entriesHash` e `previousBlockHash`.

`entriesHash`

O hash que representa todas as entradas dentro do bloco, incluindo entradas do sistema somente internas. Esse é o hash raiz da árvore Merkle, no qual os nós das folhas consistem em todos os hashes em `entriesHashList`.

`previousBlockHash`

O hash do bloco encadeado anterior no diário.

`entriesHashList`

A lista de hashes que representam cada entrada dentro do bloco. Essa lista pode incluir os seguintes hashes de entrada:

- O hash Ion que representa `transactionInfo`. Esse valor é calculado usando o hash Ion de toda a estrutura `transactionInfo`.
- Esse é o hash raiz da árvore Merkle, no qual os nós das folhas consistem em todos os hashes em `revisions`.
- O hash Ion que representa `redactionInfo`. Esse hash só existe em blocos que foram confirmados por uma transação de redação. Esse valor é calculado retirando o hash Ion de toda a estrutura `redactionInfo`.
- Hashes que representam metadados do sistema somente internos. Esses hashes podem não existir em todos os blocos.

`transactionInfo`

Uma estrutura Amazon Ion que contém informações sobre as declarações na transação que confirmou o bloqueio. Esta estrutura tem os seguintes campos:

- `statements`: a lista de instruções partiQL e `startTime` quando elas começaram a ser executadas. Cada declaração tem um hash `statementDigest`, que é necessário para calcular o hash da estrutura `transactionInfo`.
- `documents`— O documento IDs que foi atualizado pelas declarações. Cada documento inclui o `tableName` e o `tableId` ao qual pertence e o índice de cada declaração que o atualizou.

`revisions`

A lista de revisões de documentos que foram confirmadas no bloco. Cada estrutura de revisão contém todos os campos da visão comprometida da revisão.

Isso também pode incluir hashes que representam revisões internas do sistema que fazem parte de toda a cadeia de hash de um diário.

Revisões editadas

Na AmazonQLDB, uma `DELETE` declaração só exclui logicamente um documento criando uma nova revisão que o marca como excluído. QLDBtambém oferece suporte a uma operação de redação de dados que permite excluir permanentemente revisões de documentos inativas no histórico de uma tabela.

A operação de redação exclui somente os dados do usuário na revisão especificada e deixa a sequência do diário e os metadados do documento inalterados. Isso mantém a integridade geral dos dados do seu ledger. Para obter mais informações e um exemplo de uma operação de redação, consulte Redigir revisões de documentos.

Exemplo de revisão redigida

Considere o exemplo de bloco anterior. Neste bloco, suponha que você redija a revisão que tem um ID de documento `HwVFkn8IMRa0xjze5xcgga` e um número de versão de `0`.

Depois que a redação for concluída, os dados do usuário na revisão (representados pela estrutura `data`) serão substituídos por um novo campo `dataHash`. O valor desse campo é o hash de Ion da estrutura `data` removida. Como resultado, o livro mantém a integridade geral dos dados e permanece criptograficamente verificável por meio das operações de verificação existentes. API

O exemplo de revisão a seguir mostra os resultados dessa redação, com o novo `dataHash` campo destacado em *`red italics`*.

> **Note**
>
> Este exemplo de revisão é fornecido apenas para fins informativos. Os hashes mostrados não são valores reais de hash calculados.

```
...
{
  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"
  }
}
...
```

QLDBtambém acrescenta um novo bloco ao diário para a solicitação de redação concluída. Esse bloco inclui uma entrada `redactionInfo` adicional que contém uma lista de revisões que foram editadas na transação, conforme mostrado no exemplo a seguir.

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

```
    }
  ]
}
...
```

Aplicação de exemplo

Para ver um exemplo de código Java que valida a cadeia de hash de um diário usando dados exportados, consulte o GitHub repositório aws-samples/ -java. amazon-qldb-dmv-sample Esse aplicativo de exemplo inclui os seguintes arquivos de classe:

- ValidateQldbHashChain.java — Contém código de tutorial que exporta blocos de diário de um livro contábil e usa os dados exportados para validar a cadeia de hash entre os blocos.
- JournalBlock.java — Contém um método chamado `verifyBlockHash()` que demonstra como calcular cada componente de hash individual em um bloco. Esse método é chamado pelo código do tutorial em `ValidateQldbHashChain.java`.

Para obter instruções sobre como baixar e instalar esse aplicativo de amostra completo, consulte Instalando o aplicativo de amostra Amazon QLDB Java. Antes de executar o código do tutorial, siga as etapas de 1 a 3 em Tutorial de Java para configurar um ledger de amostra e carregá-lo com dados de amostra.

Consulte também

Para obter mais informações sobre periódicos emQLDB, consulte os seguintes tópicos:

- Exportação de dados de periódicos da Amazon QLDB: para aprender a exportar dados do diário para o Amazon Simple Storage Service (Amazon S3).
- Streaming de dados de diários da Amazon QLDB: para aprender a transmitir dados do diário para o Amazon Kinesis Data Streams.
- Verificação de dados na Amazon QLDB: para aprender sobre verificação criptográfica de dados do diário.

QLDBGlossário da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A seguir estão as definições dos principais termos que você pode encontrar ao trabalhar com a AmazonQLDB.

bloco | resumo | document | ID do documento | revisão do documento | entrada | Campo | índice | armazenamento indexado | diário | bloco de diário | armazenamento de diário | vertente de diário | dica de diário | ledger | prova | revisão | sessão | vertente | tabela | visualização da tabela | view

bloco

Um objeto que está comprometido com o diário em uma transação. Uma única transação grava um bloco no diário, portanto, um bloco só pode ser associado a uma transação. Um bloco que contém entradas que representam as revisões do documento que foram confirmadas na transação e as instruções partiQL que as confirmaram.

Cada bloco também tem um valor de hash para verificação. Um hash de bloco é calculado a partir dos hashes de entrada dentro desse bloco combinados com o hash do bloco encadeado anterior.

resumo

Um valor de hash de 256 bits que representa de forma exclusiva todo o histórico de revisões de documentos do seu livro em um determinado momento. Um hash de resumo é calculado a partir da cadeia de hash completa do seu diário a partir do último bloco confirmado no diário naquele momento.

QLDBpermite gerar um resumo como um arquivo de saída seguro. Em seguida, você pode usar esse arquivo de saída para verificar a integridade das revisões do documento em relação a esse hash.

document

Um conjunto de dados no formato `struct` Amazon Ion que pode ser inserido, atualizado e excluído em uma tabela. Um QLDB documento pode ter dados estruturados, semiestruturados, aninhados e sem esquema.

ID do documento

O identificador universalmente exclusivo (UUID) que é QLDB atribuído a cada documento que você insere em uma tabela. Esse ID é um número de 128 bits representado em uma sequência alfanumérica codificada em Base62 com um comprimento fixo de 22 caracteres.

revisão do documento

Uma estrutura Ion que representa uma única versão de uma sequência de documentos identificados por um ID de documento exclusivo. Uma revisão inclui seus dados de usuário (ou seja, os dados que você escreveu na tabela) e metadados gerados pelo sistema. Cada revisão está associada a uma tabela e é identificada exclusivamente por uma combinação do ID do documento e um número de versão baseado em zero.

entrada

Um objeto contido em um bloco. As entradas representam revisões de documentos que são inseridas, atualizadas e excluídas em uma transação, junto com as instruções partiQL que as confirmaram.

Cada bloco também tem um valor de hash para verificação. Um hash de entrada é calculado a partir dos hashes de revisão ou dos hashes de instrução dentro dessa entrada.

Campo

Um par nome-valor que compõe cada atributo de um QLDB documento. O nome é um token símbolo e o valor é irrestrito.

índice

Uma estrutura de dados que você pode criar em uma tabela para otimizar o desempenho das operações de recuperação de dados. Para obter informações sobre índices emQLDB, consulte CREATE INDEX na referência do Amazon QLDB PartiQL.

armazenamento indexado

O espaço em disco usado pelas tabelas, índices e histórico indexado de um ledger. O armazenamento indexado consiste em dados do ledger otimizados para consultas de alta performance.

diário

O conjunto encadeado em hash de todos os blocos que estão comprometidos em seu ledger. O diário é somente para anexar e representa um histórico completo e imutável de todas as alterações nos dados do seu ledger.

bloco de diário

Consulte bloco.

armazenamento de diário

O espaço em disco usado pelo diário de um livro contábil.

vertente de diário

Consulte vertente.

dica de diário

O último bloco confirmado em um diário em um determinado momento.

ledger

Uma instância de um recurso de banco de dados QLDB contábil da Amazon. Esse é o principal tipo de AWS recurso emQLDB. Um ledger consiste em armazenamento de diário e armazenamento indexado. Depois que os dados do ledger são enviados ao diário, eles ficam disponíveis para consulta em tabelas de revisões de documentos do Amazon Ion.

prova

A lista ordenada de valores de hash de 256 bits que QLDB retorna para um determinado resumo e revisão do documento. Ele consiste nos hashes exigidos por um modelo de árvore Merkle para encadear o hash de revisão fornecido ao hash de resumo. Você usa uma prova para verificar a integridade de suas revisões em relação ao resumo. Para ter mais informações, consulte Verificação de dados na Amazon QLDB.

revisão

Consulte revisão do documento.

sessão

Um objeto que gerencia informações sobre suas solicitações de transações de dados e respostas de e para um ledger. Uma sessão ativa (aquela que está executando ativamente uma transação) representa uma única conexão com um ledger. QLDBsuporta uma transação em execução ativa por sessão.

vertente

Uma partição de um diário. QLDBatualmente suporta periódicos com apenas uma única vertente.

tabela

Uma visão materializada de uma coleção não ordenada de revisões de documentos que são confirmadas no diário do livro contábil.

visualização da tabela

Um subconjunto consultável dos dados em uma tabela, com base nas transações confirmadas no diário. Em uma instrução partiQL, uma exibição é indicada por um qualificador de prefixo (começando com `_ql_`) para o nome de uma tabela.

Você pode consultar as seguintes visualizações definidas pelo sistema usando instruções `SELECT`:

- Usuário: a revisão ativa mais recente somente dos dados que você gravou na tabela (ou seja, o estado atual dos dados do usuário). Essa é a visualização padrão emQLDB.
- Confirmada: a revisão ativa mais recente dos dados do usuário e dos metadados gerados pelo sistema. Essa é a tabela completa definida pelo sistema que corresponde diretamente à sua tabela de usuário. Por exemplo: `_ql_committed_`*`TableName`*.

view

Consulte visualização da tabela.

Acessando a Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode acessar a Amazon QLDB usando o AWS Management Console, o AWS Command Line Interface (AWS CLI), ou QLDB API o. As seções a seguir descrevem como usar essas opções e os pré-requisitos para usá-las.

Pré-requisitos

Antes de acessarQLDB, você deve configurar um Conta da AWS se você ainda não fez isso.

Tópicos

- Inscreva-se para um Conta da AWS
- Criar um usuário com acesso administrativo
- Gerenciar QLDB permissões em IAM
- Conceder acesso programático (opcional)

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar uma.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e inserir um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário root tem acesso a todos Serviços da AWS e recursos na conta. Como prática

recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilitar AWS IAM Identity Center e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login no AWS Management Consolecomo proprietário da conta, escolhendo o usuário root e inserindo seu Conta da AWS endereço de e-mail. Na próxima página, insira sua senha.

   Para obter ajuda para fazer login usando o usuário root, consulte Como fazer login como usuário root no Início de Sessão da AWS Guia do usuário.

2. Ative a autenticação multifator (MFA) para seu usuário root.

   Para obter instruções, consulte Habilitar um MFA dispositivo virtual para seu Conta da AWS usuário root (console) no Guia do IAM usuário.

Criar um usuário com acesso administrativo

1. Ative o IAM Identity Center.

   Para obter instruções, consulte Habilitando AWS IAM Identity Center no AWS IAM Identity Center Guia do usuário.

2. No IAM Identity Center, conceda acesso administrativo a um usuário.

   Para um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como sua fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no AWS IAM Identity Center Guia do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para entrar com seu usuário do IAM Identity Center, use o login URL que foi enviado ao seu endereço de e-mail quando você criou o usuário do IAM Identity Center.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no AWS portal de acesso no Início de Sessão da AWS Guia do usuário.

Atribuir acesso a usuários adicionais

1. No IAM Identity Center, crie um conjunto de permissões que siga as melhores práticas de aplicação de permissões com privilégios mínimos.

   Para obter instruções, consulte Criar um conjunto de permissões no AWS IAM Identity Center Guia do usuário.
2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Adicionar grupos no AWS IAM Identity Center Guia do usuário.

Gerenciar QLDB permissões em IAM

Para obter informações sobre o uso AWS Identity and Access Management (IAM) para gerenciar QLDB permissões para usuários, consulteComo a Amazon QLDB trabalha com IAM.

Conceder acesso programático (opcional)

Os usuários precisam de acesso programático se quiserem interagir com AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho	Use credenciais temporárias para assinar solicitações programáticas para o AWS	Siga as instruções da interface que deseja utilizar.

Qual usuário precisa de acesso programático?	Para	Por
(Usuários gerenciados no IAM Identity Center)	CLI, AWS SDKs, ou AWS APIs.	• Para o AWS CLI, consulte Configurando o AWS CLI para usar AWS IAM Identity Center no AWS Command Line Interface Guia do usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte Autenticação do IAM Identity Center no AWS SDKsGuia de referência de ferramentas e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou AWS APIs.	Seguindo as instruções em Usando credenciais temporárias com AWS recursos no Guia do IAM usuário.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para o AWS CLI, AWS SDKs, ou AWS APIs.	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Autenticação usando credenciais de IAM usuário no AWS Command Line Interface Guia do usuário. • Para AWS SDKse ferrament as, consulte Autentica r usando credenciais de longo prazo no AWS SDKsGuia de referência de ferramentas e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para IAM usuários no Guia IAM do usuário.

Como acessar a Amazon QLDB

Depois de concluir os pré-requisitos para configurar um Conta da AWS, consulte os tópicos a seguir para saber mais sobre como acessarQLDB:

- Usar o console
- Usar o AWS CLI (APIsomente gerenciamento)
- Usando o Amazon QLDB shell (APIsomente dados)
- Usando o API

Acessando a Amazon QLDB usando o console

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode acessar o AWS Management Console para a Amazon QLDB em https://console.aws.amazon.com/qldb.

Você pode usar o console para fazer o seguinte emQLDB:

- Criar, excluir, descrever e listar ledgers.
- Execute instruções partiQL usando o editor partiQL.
- Gerencie tags para QLDB recursos.
- Verifique os dados do diário criptograficamente.
- Exporte ou transmita blocos de diários.

Para saber como criar um QLDB livro contábil da Amazon e configurá-lo com exemplos de dados do aplicativo, consulteComeçando a usar o QLDB console da Amazon.

Referência rápida do editor PartiQL

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB oferece suporte a um subconjunto do PartiQL como linguagem de consulta e do Amazon Ion como formato de dados orientado a documentos. Para obter um guia completo e informações mais detalhadas sobre a QLDB implementação do partiQL, consulte o. Referência do Amazon QLDB PartiQL

Os tópicos a seguir fornecem uma visão geral rápida de como usar o PartiQL em. QLDB

Tópicos

- Dicas rápidas do partiQL em QLDB
- Comandos
- Visualizações definidas pelo sistema.
- Regras de sintaxe básica
- Atalhos de teclado do editor PartiQL

Dicas rápidas do partiQL em QLDB

A seguir está um breve resumo das dicas e das melhores práticas para trabalhar com o PartiQL em: QLDB

- Entenda os limites de simultaneidade e transação — Todas as declarações, incluindo `SELECT` consultas, estão sujeitas a conflitos otimistas de controle de simultaneidade (OCC) e limites de transação, incluindo um tempo limite de transação de 30 segundos.
- Use índices: use índices de alta cardinalidade e execute consultas direcionadas para otimizar suas declarações e evitar a varredura completa da tabela. Para saber mais, consulte Otimizar a performance da consulta.
- Use predicados de igualdade: as pesquisas indexadas exigem um operador de igualdade (= ou `IN`). Operadores de desigualdade (<, >, `LIKE`, `BETWEEN`) não se qualificam para pesquisas indexadas e resultam em verificações de tabela completa.
- Use somente junções internas — QLDB suporta somente junções internas. Como prática recomendada, junte em campos indexados para cada tabela que você está unindo. Escolha índices de alta cardinalidade para os critérios de junção e os predicados de igualdade.

Comandos

QLDBsuporta os seguintes comandos partiQL.

Linguagem de definição de dados (DDL)

Comando	Descrição
CREATE INDEX	Cria um índice para um campo de documento de nível superior em uma tabela.
CREATE TABLE	Cria uma tabela do .
DROP INDEX	Exclua um índice de uma tabela.
DROP TABLE	Desativa uma tabela existente.
UNDROP TABLE	Reativa uma tabela inativa.

Linguagem de manipulação de dados () DML

Comando	Descrição
DELETE	Marca um documento ativo como excluído criando uma nova revisão final do documento.
FROM(INSERT,REMOVE, ouSET)	Semanticamente o mesmo que `UPDATE`.
INSERT	Adiciona um ou mais documentos a uma tabela.
SELECT	Recupera até itens de uma ou mais tabelas.
UPDATE	Atualiza, insere ou remove elementos específic os em um documento.

DMLexemplos de declarações

INSERT

```
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
}
```

UPDATE-INSERT

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

UPDATE-REMOVE

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

SELECT— Subconsulta correlacionada

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

SELECT— Junção interna

```
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— Obtenha o ID do documento usando a cláusula BY

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

Visualizações definidas pelo sistema.

QLDBsuporta as seguintes visualizações de uma tabela definidas pelo sistema.

Visão	Descrição
table_name	A visualização padrão do usuário de uma tabela que inclui somente o estado atual dos dados do usuário.
`_ql_committed_` *table_name*	A visualização confirmada completa definida pelo sistema de uma tabela que inclui o estado atual dos dados do usuário e dos metadados gerados pelo sistema, como um ID de documento.
`history(`*table_name*`)`	A função de histórico integrada que retorna o histórico completo de revisão de uma tabela.

Regras de sintaxe básica

QLDBsuporta as seguintes regras básicas de sintaxe para partiQL.

Caractere	Descrição
'	Aspas simples denotam valores de string ou nomes de campos nas estruturas Amazon Ion.
"	As aspas duplas denotam identificadores entre aspas, como uma palavra reservada usada como nome de tabela.
`	Os acentos graves denotam valores literais de Ion.
.	A notação de pontos acessa os nomes dos campos de uma estrutura principal.
`[]`	Os colchetes definem um `list` Ion ou denotam um número ordinal baseado em zero para uma lista existente.
`{}`	Chavetas definem um `struct` Ion.

Caractere	Descrição
<< >>	Os colchetes angulares duplos definem uma bolsa PartiQL, que é uma coleção não ordenada. Você usa uma sacola para inserir vários documentos em uma tabela.
Diferenciação de letras maiúsculas e minúsculas	Todos os nomes de objetos QLDB do sistema, incluindo nomes de campos e nomes de tabelas, diferenciam maiúsculas de minúsculas.

Atalhos de teclado do editor PartiQL

O editor partiQL no QLDB console oferece suporte aos seguintes atalhos de teclado.

Ação	macOS	Windows
Executar	Cmd+Return	Ctrl+Enter
Comentário	Cmd+/	Ctrl+/
Clear	Cmd+Shift+Delete	Ctrl+Shift+Delete

Acessando a Amazon QLDB usando o AWS CLI (APIsomente gerenciamento)

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode usar o AWS Command Line Interface (AWS CLI) para controlar vários Serviços da AWS a partir da linha de comando e automatize-os por meio de scripts. Você pode usar o AWS CLI para operações únicas, conforme necessário. Você também pode usá-lo para incorporar QLDB operações da Amazon em scripts utilitários.

Para CLI acessar, você precisa de um ID de chave de acesso e uma chave de acesso secreta. Use credenciais temporárias em vez de chaves de acesso de longo prazo quando possível. As credenciais temporárias incluem um ID de acesso, uma chave de acesso secreta e um token de segurança que indica quando as credenciais expiram. Para obter mais informações, consulte Usando credenciais temporárias com AWS recursos no Guia do IAM usuário.

Para obter uma lista completa e exemplos de uso de todos os comandos disponíveis QLDB no AWS CLI, veja o AWS CLI Referência de comando.

> **Note**
>
> A ferramenta AWS CLI suporta somente as API operações `qldb` de gerenciamento listadas noQLDBAPIReferência da Amazon. Isso API é usado somente para gerenciar recursos contábeis e para operações de dados não transacionais.
> Para executar transações de dados `qldb-session` API usando uma interface de linha de comando, consulteAcessando a Amazon QLDB usando o QLDB shell (APIsomente dados).

Tópicos

- Instalando e configurando o AWS CLI
- Usar o AWS CLI com QLDB

Instalando e configurando o AWS CLI

A ferramenta AWS CLI roda em Linux, macOS ou Windows. Para instalá-lo e configurá-lo, consulte as instruções a seguir no AWS Command Line Interface Guia do usuário:

1. Instalando ou atualizando a versão mais recente do AWS CLI
2. Configuração Rápida

Usar o AWS CLI com QLDB

O formato da linha de comando consiste em um nome de QLDB operação da Amazon, seguido pelos parâmetros dessa operação. A ferramenta AWS CLI suporta uma sintaxe abreviada para os valores dos parâmetros, além de. JSON

Use `help` para listar todos os comandos disponíveis emQLDB:

```
aws qldb help
```

Você também pode usar `help` para descrever um comando específico e saber mais sobre seu uso:

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

Por exemplo, para criar um ledger:

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

Acessando a Amazon QLDB usando o QLDB shell (APIsomente dados)

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon fornece um shell de linha de comando para interação com os dados transacionais. API Com o QLDB shell, você pode executar instruções partiQL em dados contábeis.

A versão mais recente desse shell é escrita em Rust e é de código aberto no GitHub repositório amazon-qldb-shellawslabs/ na ramificação padrão. `main` A versão do Python (v1) também ainda está disponível para uso no mesmo repositório na ramificação `master`.

> **Note**
>
> O QLDB shell da Amazon suporta apenas os `qldb-session` dados transacionais. API Isso API é usado somente para executar instruções partiQL em um QLDB livro contábil.
> Para interagir com as API operações `qldb` de gerenciamento usando uma interface de linha de comando, consulteAcessando a Amazon QLDB usando o AWS CLI (APIsomente gerenciamento).

Essa ferramenta não se destina a ser incorporada a um aplicativo ou adotada para fins de produção. O objetivo dessa ferramenta é permitir que você experimente rapidamente com o QLDB PartiQL.

As seções a seguir descrevem como começar a usar o QLDB shell.

Tópicos

- Pré-requisitos
- Instalar o shell
- Invocar o shell
- Parâmetros do shell
- Referência de comando
- Executar instruções individuais
- Gerenciamento de transações
- Sair do shell
- Exemplo

Pré-requisitos

Antes de começar a usar o QLDB shell, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando a Amazon QLDB. Essa transmissão inclui o seguinte:

   1. Inscreva-se em AWS.
   2. Crie um usuário com as QLDB permissões apropriadas.
   3. Conceda acesso programático para desenvolvimento.
2. Configure suas AWS credenciais e seu padrão Região da AWS. Para obter instruções, consulte Configuração rápida no Guia do usuário AWS Command Line Interface .

   Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da AWS
3. Para qualquer livro contábil no modo de `STANDARD` permissões, crie IAM políticas que concedam permissões para executar instruções partiQL nas tabelas apropriadas. Para saber como criar essas políticas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Instalar o shell

Para instalar a versão mais recente do QLDB shell, consulte o READMEarquivo.md em GitHub. QLDBfornece arquivos binários pré-criados para Linux, macOS e Windows na seção Versões do repositório. GitHub

Para macOS, o shell se integra ao `aws/tap` Homebrew tap. Para instalar o shell no macOS usando Homebrew, execute os comandos a seguir.

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

Configuração

Após a instalação, o shell carrega o arquivo de configuração padrão localizado em `$XDG_CONFIG_HOME/qldbshell/config.ion` durante a inicialização. No Linux e macOS, esse arquivo está normalmente em `~/.config/qldbshell/config.ion`. Se esse arquivo não existir, o shell será executado com as configurações padrão.

Você pode criar um arquivo `config.ion` manualmente após a instalação. Esse arquivo de configuração usa o formato de dados Amazon Ion. Este é um exemplo de um arquivo mínimo `config.ion`.

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

Se `default_ledger` não estiver definido no seu arquivo de configuração, o parâmetro `--ledger` será necessário quando você invocar o shell. Para obter uma lista completa das opções de configuração, consulte o READMEarquivo.md em GitHub.

Invocar o shell

Para invocar o QLDB shell em seu terminal de linha de comando para um livro contábil específico, execute o comando a seguir. Substituir *`my-example-ledger`* com o nome do seu livro contábil.

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

Esse comando se conecta ao seu padrão Região da AWS. Para especificar explicitamente a Região, você pode executar o comando com o parâmetro `--region` ou `--qldb-session-endpoint`, conforme descrito na seção a seguir.

Depois de invocar uma sessão de shell `qldb`, você pode inserir os seguintes tipos de entrada:

- Comandos do Shell
- Instruções partiQL únicas em transações separadas
- Várias instruções partiQL em uma transação

Parâmetros do shell

Para obter uma lista completa dos sinalizadores e opções disponíveis para invocar um shell, execute o comando `qldb` com o sinalizador `--help`, da seguinte maneira.

```
$ qldb --help
```

Veja a seguir alguns sinalizadores principais e opções para o comando `qldb`. Você pode adicionar esses parâmetros opcionais para substituir o perfil de credenciais Região da AWS, o endpoint, o formato dos resultados e outras opções de configuração.

Uso

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

FLAGS

`-h`, `--help`

Imprime informações de ajuda.

`-v`, `--verbose`

Configura a verbosidade do log. Por padrão, o shell registra somente erros. Para aumentar o nível de verbosidade, repita esse argumento (por exemplo,`-vv`). O nível mais alto é `-vvv`, o que corresponde à verbosidade `trace`.

`-V`, `--version`

Imprime as informações da versão.

OPTIONS

`-l`, **`--ledger`** ***`LEDGER_NAME`***

É o nome do ledger ao qual se conectar. Esse é um parâmetro de shell obrigatório se `default_ledger` não estiver definido em seu arquivo `config.ion`. Nesse arquivo, você pode definir opções adicionais, como a Região.

`-c`, **`--config`** ***`CONFIG_FILE`***

O arquivo em que você pode definir qualquer opção de configuração do shell. Para obter detalhes de formatação e uma lista completa das opções de configuração, consulte o READMEarquivo.md em. GitHub

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

O formato de saída dos resultados da sua consulta. O padrão é `ion`.

`-p`, **`--profile`** ***`PROFILE`***

A localização do seu perfil de AWS credenciais a ser usado para autenticação.

Se não for fornecido, o shell usa seu AWS perfil padrão, localizado em`~/.aws/credentials`.

`-r`, **`--region`** ***`REGION_CODE`***

O Região da AWS código do QLDB livro contábil ao qual se conectar. Por exemplo: `us-east-1`.

Se não for fornecido, o shell se conectará ao seu padrão Região da AWS conforme especificado em seu AWS perfil.

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

O `qldb-session` API endpoint ao qual se conectar.

Para obter uma lista completa das QLDB regiões e endpoints disponíveis, consulte os QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Referência de comando

Depois de invocar uma sessão `qldb`, o shell oferece suporte às seguintes chaves e comandos de banco de dados:

Chaves Shell

Chave	Descrição da função
Enter	Executa a instrução.
Escape+Enter (macOS, Linux) Shift+Enter (Windows)	Inicia uma nova linha para inserir uma declaração que abrange várias linhas. Você também pode copiar o texto de entrada com várias linhas e colá-lo no shell. Para obter instruções sobre como configurar Option em vez de Escape como chave meta no macOS, consulte o site do OS X Daily.
Ctrl+C	Cancela o comando atual.
Ctrl+D	Sinaliza o fim do arquivo (EOF) e sai do nível atual do shell. Se não estiver em uma transação ativa, sai do shell. Em uma transação ativa, aborta a transação.

Comandos do banco de dados do shell

Comando	Descrição da função
`help`	Exibe as informações de ajuda.
`begin`	Inicia uma transação.
`start transaction`	
`commit`	Confirma sua transação no diário do ledger.
`abort`	Interrompe sua transação e rejeita todas as alterações que você fez.
`exit`	Sai do shell.
`quit`	

> **Note**
>
> Todos os comandos QLDB do shell não diferenciam maiúsculas de minúsculas.

Executar instruções individuais

Com exceção dos comandos do banco de dados e dos meta-comandos do shell listados em README.md, o shell interpreta cada comando inserido como uma instrução partiQL separada. Por padrão, o shell ativa o modo `auto-commit`. Esse modo é configurável.

No modo `auto-commit`, o shell executa implicitamente cada instrução em sua própria transação e confirma automaticamente a transação se nenhum erro for encontrado. Isso significa que você não precisa executar `start transaction` (ou `begin`) e `commit` manualmente toda vez que executar uma instrução.

Gerenciamento de transações

Como alternativa, o QLDB shell permite que você controle manualmente as transações. Você pode executar várias instruções em uma transação de forma interativa ou não interativa, agrupando comandos e instruções em lote sequencialmente.

Transações interativas

Para executar uma transação interativa, execute as etapas a seguir.

1. Para iniciar uma transação, digite o comando `begin`.

   ```
   qldb> begin
   ```

   Depois de iniciar uma transação, o shell exibirá o seguinte prompt de comando.

   ```
   qldb *>
   ```

2. Em seguida, cada instrução inserida é executada na mesma transação.
   - Por exemplo, é possível executar uma única instrução da seguinte forma.

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

Depois de pressionar Enter, o shell exibe os resultados da instrução.

- Você também pode inserir várias instruções ou comandos separados por um delimitador de ponto e vírgula (`;`) da seguinte forma.

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

3. Para finalizar a transação, insira um dos seguintes comandos.
   - Insira o comando `commit` para confirmar sua transação no diário do ledger.

```
qldb *> commit
```

   - Digite o comando `abort` para interromper sua transação e rejeitar as alterações feitas.

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

Tempo limite de transação

Uma transação interativa segue o limite de tempo limite QLDB da transação. Se você não confirmar uma transação dentro de 30 segundos após iniciá-la, a transação QLDB expirará automaticamente e rejeitará todas as alterações feitas durante a transação.

Então, em vez de exibir os resultados da instrução, o shell exibe uma mensagem de erro de expiração e retorna ao prompt de comando normal. Para tentar novamente, você deve digitar o comando `begin` novamente para iniciar uma nova transação.

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

Transações não-interativas

Você pode executar uma transação completa com várias instruções agrupando comandos em lotes e instruções sequencialmente da seguinte maneira.

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

Você deve separar cada comando e instrução com um delimitador de ponto e vírgula (`;`). Se alguma instrução na transação não for válida, o shell rejeitará automaticamente a transação. O shell não prossegue com nenhuma instrução subsequente que você inseriu.

Também é possível configurar várias transações.

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

Semelhante ao exemplo anterior, se uma transação falhar, o shell não prosseguirá com nenhuma transação ou instrução subsequente que você inseriu.

Se você não finalizar uma transação, o shell mudará para o modo interativo e solicitará o próximo comando ou instrução.

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

Sair do shell

Para sair da sessão atual do shell `qldb`, digite o comando `exit` ou `quit` ou use o atalho de teclado Ctrl + D quando o shell não estiver em uma transação.

```
qldb> exit
$
```

```
qldb> quit
$
```

Exemplo

Para obter informações sobre como escrever instruções partiQL emQLDB, consulte o. Referência do Amazon QLDB PartiQL

Example

O exemplo a seguir exibe uma sequência comum de comandos básicos.

> **Note**
>
> O QLDB shell executa cada instrução partiQL neste exemplo em sua própria transação.

Este exemplo pressupõe que o ledger `test-ledger` já exista e esteja ativo.

```
$ 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
```

Acessando a Amazon QLDB usando o API

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode usar o AWS Management Console e o AWS Command Line Interface (AWS CLI) para trabalhar de forma interativa com a AmazonQLDB. No entanto, para aproveitar ao máximoQLDB, você pode escrever o código do aplicativo com um QLDB driver ou um AWS SDKpara interagir com seu livro contábil usando o. APIs

O driver permite que seu aplicativo interaja com o QLDB uso dos dados transacionais. API A ferramenta AWS SDKsuporta a interação com o gerenciamento de QLDB recursosAPI. Para obter mais informações sobre elesAPIs, consulte QLDBAPIReferência da Amazon o.

O driver fornece suporte para QLDB em Java,. NET, Go, Node.js e Python. Para começar a trabalhar rapidamente com essas linguagens, consulte Começando com o QLDB driver da Amazon.

Antes que você possa usar um QLDB driver ou um AWS SDKem seu aplicativo, você deve conceder acesso programático. Para obter mais informações, consulte Conceder acesso programático.

Começando a usar o QLDB console da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este tutorial orienta você nas etapas para criar seu primeiro QLDB livro contábil da Amazon e preenchê-lo com tabelas e dados de amostra. O exemplo de livro contábil que você cria nesse cenário é um banco de dados para um aplicativo do departamento de veículos motorizados (DMV) que rastreia as informações históricas completas sobre registros de veículos.

O histórico de um ativo é um caso de uso comum QLDB porque envolve uma variedade de cenários e operações que destacam a utilidade de um banco de dados contábil. Com ele, QLDB você pode acessar, consultar e verificar diretamente o histórico completo das alterações em seus dados em um banco de dados orientado a documentos que oferece suporte a recursos de consulta SQL semelhantes aos da.

À medida que você trabalha neste tutorial, os tópicos a seguir explicam como adicionar registros de veículos, modificá-los e visualizar o histórico de alterações nesses registros. Este guia também mostra como verificar criptograficamente um documento de registro e conclui limpando os recursos e excluindo o ledger de amostra.

Cada etapa do tutorial tem instruções para usar o AWS Management Console.

Tópicos

- Tutorial de pré-requisitos e considerações
- Etapa 1: criar um novo ledger
- Etapa 2: criar tabelas, índices e dados de amostra em um ledger
- Etapa 3: consultar as tabelas em um ledger
- Etapa 4: Modificar documentos em um ledger
- Etapa 5: Visualizar o histórico de revisão de um documento
- Etapa 6: verificar um documento em um ledger

- Etapa 7 (opcional): Limpar os recursos
- Começando com a AmazonQLDB: próximas etapas

Tutorial de pré-requisitos e considerações

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Antes de começar este QLDB tutorial da Amazon, certifique-se de preencher os seguintes pré-requisitos:

1. Siga as instruções de AWS configuração emAcessando a Amazon QLDB, caso ainda não tenha feito isso. Essas etapas incluem a inscrição AWS e a criação de um usuário administrativo.
2. Siga as instruções em Configurar permissões para configurar IAM permissões para seus QLDB recursos. Para concluir todas as etapas neste tutorial, você precisa de acesso administrativo total aos recursos do ledger por meio do AWS Management Console.

   > **Note**
   >
   > Se você já estiver conectado como um usuário com permissões AWS administrativas completas, você pode pular esta etapa.

3. (Opcional) QLDB criptografa dados em repouso usando uma chave em AWS Key Management Service (AWS KMS). Escolha um dos seguintes tipos de AWS KMS keys:
   - AWS KMSchave de propriedade — Use uma KMS chave de propriedade e gerenciada AWS em seu nome. Essa é a opção padrão e não requer configuração adicional.
   - KMSChave gerenciada pelo cliente — use uma KMS chave de criptografia simétrica em sua conta que você cria, possui e gerencia. QLDBnão suporta chaves assimétricas.

     Essa opção exige que você crie uma KMS chave ou use uma chave existente na sua conta. Para obter instruções sobre como criar uma chave gerenciada pelo cliente, consulte Criação de KMS chaves de criptografia simétricas no Guia do AWS Key Management Service desenvolvedor.

Você pode especificar uma KMS chave gerenciada pelo cliente usando um ID, alias ou Amazon Resource Name (ARN). Para saber mais, consulte Identificadores de chave (KeyId) no Guia do AWS Key Management Service desenvolvedor.

> **Note**
>
> Chaves entre regiões não são compatíveis. A KMS chave especificada deve estar na Região da AWS mesma do seu livro contábil.

Configurar permissões

Nesta etapa, você configura as permissões de acesso total por meio do console para todos os QLDB recursos do seu Conta da AWS. Para conceder essas permissões rapidamente, use a política AWS gerenciada mazonQLDBConsole FullAccess A.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .
- Usuários gerenciados IAM por meio de um provedor de identidade:

  Crie um perfil para a federação de identidades. Siga as instruções em Criação de uma função para um provedor de identidade terceirizado (federação) no Guia IAM do usuário.
- IAMusuários:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de uma função para um IAM usuário no Guia IAM do usuário.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adicionar permissões a um usuário (console) no Guia do IAM usuário.

> **Important**
>
> Para os fins deste tutorial, você concede a si mesmo acesso administrativo total a todos os QLDB recursos. Para casos de uso de produção, no entanto, siga a prática recomendada

de segurança de concessão de privilégio mínimo ou conceda apenas as permissões necessárias para realizar uma tarefa. Para ver exemplos, consulte Exemplos de políticas baseadas em identidade para a Amazon QLDB.

Para criar um ledger chamado `vehicle-registration`, vá para Etapa 1: criar um novo ledger.

Etapa 1: criar um novo ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você cria um novo QLDB livro contábil da Amazon chamado`vehicle-registration`. Em seguida, você confirma que o status do livro é Ativo. Você também pode verificar todas as tags adicionadas ao ledger.

Ao criar um ledger, a proteção contra exclusão é habilitada, por padrão. A proteção contra exclusão é um recurso QLDB que impede que os livros contábeis sejam excluídos por qualquer usuário. Você pode desativar a proteção contra exclusão ao criar um livro contábil usando o QLDB API ou o AWS Command Line Interface ()AWS CLI.

Para criar um novo ledger

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Conceitos básicos.
3. Em Criar seu primeiro ledger, escolha Criar ledger.
4. Na página Criar Ledger, faça o seguinte:
   - Informações do ledger: o nome do ledger deve ser pré-preenchido com **`vehicle-registration`**.
   - Modo de permissões: o modo de permissões a ser atribuído ao ledger. Escolha uma das seguintes opções:

- Permitir tudo — Um modo de permissões antigo que permite o controle de acesso com granularidade API de nível para livros contábeis.

  Esse modo permite que os usuários que têm `SendCommand` API permissão para esse registro executem todos os comandos do PartiQL (portanto`ALLOW_ALL`,) em qualquer tabela no livro contábil especificado. Esse modo ignora qualquer política de IAM permissões em nível de tabela ou de comando que você cria para o livro contábil.

- Padrão - (Recomendado) Um modo de permissões que permite o controle do acesso com granularidade mais fina para ledgers, tabelas e comandos PartiQL. Recomendamos o uso deste modo de permissões para maximizar a segurança dos dados do seu ledger.

  Por padrão, esse modo nega todas as solicitações de execução de comandos do PartiQL em qualquer tabela nesse ledger. Para permitir comandos do PartiQL, você deve criar políticas de IAM permissões para recursos específicos da tabela e ações do PartiQL, além da `SendCommand` API permissão para o livro contábil. Para obter mais informações, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

- Criptografar dados em repouso: a chave em AWS Key Management Service (AWS KMS) a ser usada para criptografia de dados em repouso. Escolha uma das seguintes opções:
  - Use uma KMS chave AWS própria — Use uma KMS chave que seja de sua propriedade e gerenciada AWS em seu nome. Essa é a opção padrão e não requer configuração adicional.
  - Escolha uma AWS KMS chave diferente — use uma KMS chave de criptografia simétrica em sua conta que você cria, possui e gerencia.

    Para criar uma nova chave usando o AWS KMS console, escolha Criar uma AWS KMS chave. Para obter mais informações, consulte Criação de KMS chaves de criptografia simétricas no Guia do AWS Key Management Service desenvolvedor.

    Para usar uma KMS chave existente, escolha uma na lista suspensa ou especifique uma KMS chave. ARN

- Tags: (optional) adicionar metadados à função anexando etiquetas como pares de chave-valor. Você pode adicionar tags ao seu ledger para ajudar a organizá-lo e identificá-lo. Para ter mais informações, consulte Marcação de recursos da Amazon QLDB.

  Escolha Adicionar tag e, em seguida, insira os pares de valores-chave, conforme apropriado.

5. Quando estiver satisfeito com as configurações, escolha Criar ledger.

> **Note**
>
> Você pode acessar seu QLDB livro contábil quando seu status se tornar Ativo. Isso pode demorar vários minutos.

6. Na lista de Ledgers, localize `vehicle-registration` e confirme se o status do livro está Ativo.
7. (Opcional) Escolha o nome do ledger `vehicle-registration`. Na página de detalhes do ledger de registro do veículo, confirme se todas as tags que você adicionou ao livro aparecem no cartão Tags. Você também pode editar suas tags do ledger usando esta página do console.

Para criar tabelas no ledger `vehicle-registration`, vá para Etapa 2: criar tabelas, índices e dados de amostra em um ledger.

Etapa 2: criar tabelas, índices e dados de amostra em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Quando seu QLDB livro contábil da Amazon está ativo, você pode começar a criar tabelas para dados sobre veículos, seus proprietários e suas informações de registro. Depois de criar as tabelas e os índices, você pode carregá-los com dados.

Nesta etapa, você cria quatro tabelas no ledger `vehicle-registration`:

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

Você também cria os índices a seguir.

Nome da tabela	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicensePlateNumber`
`DriversLicense`	`PersonId`

Você pode usar o QLDB console para criar automaticamente essas tabelas com índices e carregá-las com dados de amostra. Ou você pode usar o editor partiQL no console para executar manualmente cada instrução partiQL. step-by-step

Opção automática

Para criar tabelas, índices e dados de amostra

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Conceitos básicos.
3. Na opção Automático no cartão de dados do aplicativo de amostra, escolha `vehicle-registration` na lista de ledgers.
4. Escolha Carregar dados de exemplo.

   Se a operação for concluída com êxito, o console exibirá a mensagem Dados de amostra carregados.

   Esse script executa todas as instruções em uma única transação. Se alguma parte da transação falhar, cada instrução será revertida e uma mensagem de erro apropriada será exibida. Você pode repetir a operação depois de resolver qualquer problema.

> **Note**
>
> - Uma possível causa para uma falha na transação é a tentativa de criar tabelas duplicadas. Sua solicitação para carregar dados de exemplo falhará se algum dos seguintes nomes de tabela já existir em seu ledger: `VehicleRegistration`, `Vehicle`,`Person`, e `DriversLicense`.
>
> Em vez disso, tente carregar esses dados de amostra em um ledger vazio.
> - Esse script executa instruções `INSERT` parametrizadas. Portanto, essas instruções partiQL são registradas em seus blocos de diário com parâmetros de associação em vez de dados literais. Por exemplo, você pode ver a seguinte instrução em um bloco de diário, em que o ponto de interrogação (`?`) é um marcador variável para o conteúdo do documento.
>
>  ```
>  INSERT INTO Vehicle ?
>  ```

Opção manual

Você insere documentos em `VehicleRegistration` com um campo `PrimaryOwner` vazio e em `DriversLicense` com um campo `PersonId` vazio. Posteriormente, você preenche esses campos com o documento `id`atribuído pelo sistema da tabela `Person`.

> **Tip**
>
> Como prática recomendada, use esse campo de metadados `id` do documento como uma chave estrangeira. Para ter mais informações, consulte Consultar metadados do documento.

Para criar tabelas, índices e dados de amostra

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, selecione Editor PartiQL.
3. Escolha o ledger `vehicle-registration`.
4. Comece criando quatro tabelas. QLDBoferece suporte a conteúdo aberto e não impõe esquema, portanto, você não especifica atributos ou tipos de dados.

Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar. Para executar a instrução, você também pode usar o atalho de teclado Ctrl+Enter para Windows ou Cmd+Return para macOS. Para obter mais atalhos de teclado, consulte Atalhos de teclado do editor PartiQL.

```
CREATE TABLE VehicleRegistration
```

Repita esta etapa para cada uma das opções a seguir.

```
CREATE TABLE Vehicle
```

```
CREATE TABLE Person
```

```
CREATE TABLE DriversLicense
```

5. Em seguida, crie índices que otimizem o desempenho da consulta para cada tabela.

> **Important**
>
> QLDBrequer um índice para pesquisar um documento com eficiência. Sem um índice, QLDB precisa fazer uma digitalização completa da tabela ao ler documentos. Isso pode causar problemas de desempenho em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade (= ou `IN`) em um campo indexado ou em um ID de documento. Para ter mais informações, consulte Otimizar a performance da consulta.

Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

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

Repita esta etapa para o seguinte.

```
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. Depois de criar seus índices, você pode começar a carregar dados em suas tabelas. Nesta etapa, insira documentos na tabela `Person` com informações pessoais sobre os proprietários dos veículos que o livro está rastreando.

   Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

```
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. Em seguida, preencha a tabela `DriversLicense` com documentos que incluam informações da carteira de motorista de cada proprietário do veículo.

   Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

```
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. Agora, preencha a tabela `VehicleRegistration` com os documentos de registro do veículo. Esses documentos incluem uma estrutura `Owners` aninhada que armazena os proprietários primários e secundários.

   Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

```
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. Por fim, preencha a tabela `Vehicle` com documentos que descrevem os veículos que estão registrados em seu ledger.

   Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

```
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'
} >>
```

Em seguida, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`. Vá para Etapa 3: consultar as tabelas em um ledger.

Etapa 3: consultar as tabelas em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de criar tabelas em um QLDB livro contábil da Amazon e carregá-las com dados, você pode executar consultas para revisar os dados de registro do veículo que você acabou de inserir. QLDBusa o PartiQL como sua linguagem de consulta e o Amazon Ion como seu modelo de dados orientado a documentos.

O partiQL é uma linguagem de consulta de código aberto SQL compatível que foi estendida para funcionar com o Ion. Com o partiQL, você pode inserir, consultar e gerenciar seus dados com operadores conhecidosSQL. Amazon Ion é um superconjunto de. JSON O Ion é um formato de dados de código aberto baseado em documentos que oferece a flexibilidade de armazenar e processar dados estruturados, semiestruturados e aninhados.

Nesta etapa, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`.

> **Warning**
>
> Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. O partiQL suporta essas consultas porque é compatível. SQL No entanto, não execute varreduras de tabela para casos de uso de produção emQLDB.

Verificações de tabela podem causar problemas de performance em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Para ter mais informações, consulte Otimizar a performance da consulta.

Para consultar as tabelas

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, selecione Editor PartiQL.
3. Escolha o ledger `vehicle-registration`.
4. Na janela do editor de consultas, insira a seguinte instrução para consultar a `Vehicle` tabela de um determinado número de identificação do veículo (VIN) que você adicionou ao livro contábil e, em seguida, escolha Executar.

   Para executar a instrução, você também pode usar o atalho de teclado Ctrl+Enter para Windows ou Cmd+Return para macOS. Para obter mais atalhos de teclado, consulte Atalhos de teclado do editor PartiQL.

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

5. Você pode escrever consultas de junção internas. Este exemplo de consulta une `Vehicle` a `VehicleRegistration` e retorna as informações de registro junto com os atributos do veículo registrado para um determinado `VIN`.

   Insira a instrução a seguir e escolha Executar.

   ```
   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
   ```

   Você também pode unir as tabelas `Person` e `DriversLicense` para ver os atributos relacionados aos motoristas que foram adicionados ao ledger.

Repita esta etapa para o seguinte.

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

Para saber mais sobre a modificação de documentos nas tabelas do ledger `vehicle-registration`, consulte Etapa 4: Modificar documentos em um ledger.

Etapa 4: Modificar documentos em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Agora que você tem dados com os quais trabalhar, pode começar a fazer alterações nos documentos do `vehicle-registration` livro contábil na AmazonQLDB. Por exemplo, considere o Audi A5 com. VIN `1N4AL11D75C109151` Este carro é inicialmente de propriedade de um motorista chamado Raul Lewis em Seattle, WA.

Suponha que Raul venda o carro para um residente em Everett, WA chamado Brent Logan. Então, Brent e Alexis Pena decidem se casar. Brent quer adicionar Alexis como proprietária secundária no registro. Nesta etapa, as seguintes instruções de linguagem de manipulação de dados (DML) demonstram como fazer as alterações apropriadas em seu livro contábil para refletir esses eventos.

> **Tip**
>
> Como prática recomendada, use o `id` de um documento atribuído pelo sistema como uma chave estrangeira. Embora você possa definir campos destinados a serem identificadores exclusivos (por exemplo, de um veículoVIN), o verdadeiro identificador exclusivo de um documento é seu`id`. Esse campo está incluído nos metadados do documento, que você pode consultar na visualização confirmada (a visualização definida pelo sistema de uma tabela).

Para obter mais informações sobre exibições emQLDB, consulteConceitos principais. Para saber mais sobre metadados, consulte Consultar metadados do documento.

Para modificar documentos

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, selecione Editor PartiQL.
3. Escolha o ledger `vehicle-registration`.

   > **Note**
   >
   > Se você configurar seu ledger usando o atributo automático Carregar dados de exemplo do console, vá para a etapa 6.

4. Se você executou as instruções `INSERT` manualmente para carregar os dados de amostra, continue com essas etapas.

   Para registrar inicialmente Raul como proprietário desse veículo, comece encontrando o documento atribuído pelo sistema `id` na tabela `Person`. Esse campo está incluído nos metadados do documento, que você pode consultar na visualização definida pelo sistema de uma tabela, chamada visualização confirmada.

   Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

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

   O prefixo `_ql_committed_` é um prefixo reservado que significa que você deseja consultar a exibição confirmada da tabela `Person`. Nessa exibição, seus dados estão aninhados no campo `data` e os metadados estão aninhados no campo `metadata`.

5. Agora, use esse `id` em uma instrução `UPDATE` para modificar o documento apropriado na tabela `VehicleRegistration`. Insira a instrução a seguir e escolha Executar.

   ```
   UPDATE VehicleRegistration AS r
   SET r.Owners.PrimaryOwner.PersonId = '294jJ3YUoH1IEEm8GSab0s' --replace with your
    id
   WHERE r.VIN = '1N4AL11D75C109151'
   ```

Confirme se você modificou o campo `Owners` emitindo essa instrução.

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

6. Para transferir a propriedade do veículo para Brent, na cidade de Everett, primeiro encontre sua `id` na tabela `Person` com a seguinte instrução.

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

Em seguida, use essa `id` para atualizar `PrimaryOwner` e `City` na tabela `VehicleRegistration`.

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

Confirme se você modificou os campos `PrimaryOwner` e `City` emitindo essa instrução.

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

7. Para adicionar Alexis como proprietária secundária do carro, encontre seu `Person id`.

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

Em seguida, `id` insira isso na `SecondaryOwners` lista com a seguinte INSERT DML declaração FROM-.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners
    VALUE { 'PersonId' : '5Ufgdlnj06gF5CWcOIu64s' } --replace with your id
```

Confirme se você modificou o `SecondaryOwners` emitindo essa instrução.

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

Para revisar essas alterações no ledger `vehicle-registration`, consulte Etapa 5: Visualizar o histórico de revisão de um documento.

Etapa 5: Visualizar o histórico de revisão de um documento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de modificar os dados de registro do carro com VIN `1N4AL11D75C109151`, você pode consultar o histórico de todos os proprietários registrados e quaisquer outros campos atualizados. Você pode ver todas as revisões de um documento que inseriu, atualizou e excluiu consultando a Função de histórico incorporada.

A função de histórico retorna revisões da visualização confirmada de sua tabela, que inclui os dados do aplicativo e os metadados associados. Os metadados mostram exatamente quando cada revisão foi feita, em que ordem e qual transação as confirmou.

Nesta etapa, você consulta o histórico de revisão de um documento na tabela `VehicleRegistration` do ledger `vehicle-registration`.

Visualizar o histórico de revisão

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, selecione Editor PartiQL.
3. Escolha o ledger `vehicle-registration`.
4. Para consultar o histórico de um documento, comece encontrando seu `id` único. Além de consultar a visualização confirmada, outra forma de obter um documento `id` é usar a palavra-

chave BY na visualização padrão do usuário da tabela. Para saber mais, consulte Usar a cláusula BY para consultar o ID do documento.

Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

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

5. Em seguida, você pode usar esse valor `id` para consultar a função de histórico. Insira a instrução a seguir e escolha Executar. Certifique-se de substituir o valor `id` pelo seu próprio ID do documento, conforme apropriado.

```
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**
>
> Para os fins deste tutorial, essa consulta de histórico retorna todas as revisões do ID `ADR2LQq48kB9neZDupQrMm` do documento. Como prática recomendada, no entanto, qualifique uma consulta de histórico com um ID do documento e um intervalo de datas (hora de início e hora de término).
> EmQLDB, cada `SELECT` consulta é processada em uma transação e está sujeita a um limite de tempo limite de transação. As consultas de histórico que incluem uma hora de início e uma hora de término ganham o benefício da qualificação por intervalo de datas. Para ter mais informações, consulte Função de histórico.

A função de histórico retorna documentos no mesmo esquema da visualização confirmada. Este exemplo projeta seus dados de registro de veículos modificados. A saída deve ser semelhante à seguinte.

VIN	Cidade	Proprietários
`"1N4AL11D 75C109151"`	`"Seattle"`	`{PrimaryOwner:{PersonId:""} ,SecondaryOwners:[]}`

VIN	Cidade	Proprietários
"1N4AL11D 75C109151"	"Seattle"	{PrimaryOwner:{PersonId:"29 4jJ3YUoH1IEEm8GSabOs"}, SecondaryOwners:[]}
"1N4AL11D 75C109151"	"Everett"	{PrimaryOwner:{PersonId:"7N mE8YLPbXc0IqesJy1rpR"}, SecondaryOwners:[]}
"1N4AL11D 75C109151"	"Everett"	{PrimaryOwner:{PersonId:"7N mE8YLPbXc0IqesJy1rpR"}, SecondaryOwners:[{PersonId: "5Ufgdlnj06gF5CWcOIu64s"}]}

> **Note**
>
> A consulta do histórico pode nem sempre retornar as revisões do documento em ordem sequencial.

Analise a saída e confirme se as alterações refletem o que você fez em Etapa 4: Modificar documentos em um ledger.

6. Em seguida, você pode inspecionar os metadados do documento de cada revisão. Insira a instrução a seguir e escolha Executar. Novamente, certifique-se de substituir o valor `id` pelo seu próprio ID do documento, conforme apropriado.

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

A saída deve ser semelhante à seguinte.

versão	id	txTime	txId
0	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T19:20:360d-3Z	"FMoVdWuPxJg3k466Iz4i75"
1	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:40:199d-3Z	"KWByxe842Xw8DNHcvARPOt"
2	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:44:432d-3Z	"EKwDOJRwbHpFvmAyJ2Kdh9"
3	"ADR2LQq48kB9neZDupQrMm"	2019-05-23T21:49:254d-3Z	"96EiZd7vCmJ6RAvOvTZ4YA"

Esses campos de metadados fornecem detalhes sobre quando cada item foi modificado e por qual transação. A partir desses dados, você pode deduzir o seguinte:

- O documento é identificado exclusivamente por seu `id` atribuído pelo sistema: `ADR2LQq48kB9neZDupQrMm`. Esse é um identificador universalmente exclusivo (UUID) representado em uma string codificada em Base62.
- O `txTime` mostra que a revisão inicial do documento (versão `0`) foi criada em `2019-05-23T19:20:360d-3Z`.
- Cada transação subsequente cria uma nova revisão com o mesmo documento `id`, um número de versão incrementado e `txId` e `txTime` atualizados.

Para verificar uma revisão de documento criptograficamente no ledger `vehicle-registration`, vá para Etapa 6: verificar um documento em um ledger.

Etapa 6: verificar um documento em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Com a AmazonQLDB, você pode verificar com eficiência a integridade de um documento no diário do seu livro contábil usando hash criptográfico com -256. SHA Neste exemplo, Alexis e Brent decidem fazer o upgrade para um novo modelo trocando o veículo com uma VIN `1N4AL11D75C109151` concessionária de automóveis. A concessionária inicia o processo verificando a propriedade do veículo no cartório.

Para saber mais sobre como a verificação e o hashing criptográfico funcionam emQLDB, consulte. Verificação de dados na Amazon QLDB

Nesta etapa, você verifica uma revisão do documento no ledger `vehicle-registration`. Primeiro, você solicita um resumo, que é retornado como um arquivo de saída e atua como uma assinatura de todo o histórico de alterações do seu ledger. Em seguida, você solicita uma prova da revisão em relação a esse resumo. Usando essa prova, a integridade da sua revisão é verificada se todas as verificações de validação forem aprovadas.

Para solicitar um resumo

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista de ledgers, selecione `vehicle-registration`.
4. Escolha Obter resumo. A caixa de diálogo Obter resumo exibe os seguintes detalhes do resumo:
   - Digest — O valor de hash SHA -256 do resumo que você solicitou.
   - Endereço da dica de resumo: a última localização do bloco no diário coberto pelo resumo que você solicitou. Um endereço tem os dois campos a seguir:
     - `strandId`: o ID exclusivo da cadeia do diário que contém o bloco.
     - `sequenceNo`: o número do índice que especifica a localização do bloco dentro da cadeia.
   - ledger: o nome do ledger para o qual você solicitou um resumo.
   - Data: a data e hora em que você solicitou o resumo.
5. Analise as informações do resumo. Em seguida, escolha Salvar. Você pode manter o nome do arquivo padrão ou inserir um novo nome.

   Essa etapa salva um arquivo de texto simples com conteúdo no formato Amazon Ion. O arquivo tem uma extensão de nome de arquivo `.ion.txt` e contém todas as informações resumidas listadas na caixa de diálogo anterior. Este é um exemplo do conteúdo de um arquivo de resumo: A ordem dos campos pode variar dependendo do seu navegador.

```
{
  "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
  "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
  "ledger": "vehicle-registration",
  "date": "2019-04-17T16:57:26.749Z"
}
```

6. Salve esse arquivo onde você possa acessá-lo mais tarde. Nas etapas a seguir, você usa esse arquivo para verificar a revisão de um documento.

Depois de salvar um resumo de ledger, você pode iniciar o processo de verificação de uma revisão do documento em relação a esse resumo.

> **Note**
>
> Em um caso de uso de produção para verificação, você usa um resumo que foi salvo anteriormente em vez de realizar as duas tarefas consecutivamente. Como prática recomendada, solicite e salve o resumo assim que uma revisão que você deseja verificar posteriormente for gravada no diário.

Para verificar a revisão de um documento

1. Primeiro, consulte seu ledger para saber o `id` e `blockAddress` da revisão do documento que você deseja verificar. Esses campos estão incluídos nos metadados do documento, que você pode consultar na visualização confirmada.

   O documento `id` é uma sequência de caracteres de identificação exclusiva atribuída pelo sistema. `blockAddress` é uma estrutura Ion que especifica a localização do bloco onde a revisão foi confirmada.

   No painel de navegação do QLDB console, escolha editor partiQL.

2. Escolha o ledger `vehicle-registration`.
3. Na janela do editor de consulta, insira a instrução a seguir e, em seguida, escolha Executar.

```
SELECT r.metadata.id, r.blockAddress
FROM _ql_committed_VehicleRegistration AS r
```

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

4. Copie e salve os valores `id` e `blockAddress` que sua consulta devolve. Certifique-se de omitir as aspas duplas do campo `id`. No Amazon Ion, os tipos de dados de string são delimitados com aspas duplas.
5. Agora que você selecionou uma revisão do documento, pode iniciar o processo de verificação.

   No painel de navegação, escolha Verificação.
6. No formulário Verificar documento, em Especificar o documento que deseja verificar, insira os seguintes parâmetros de entrada:

   - Ledger: escolha `vehicle-registration`.
   - Endereço do bloco: o valor `blockAddress` devolvido por sua consulta na etapa 3.
   - ID do documento: o valor `id` devolvido por sua consulta na etapa 3.
7. Em Especificar o resumo a ser usado para verificação, selecione o resumo que você salvou anteriormente escolhendo Escolher resumo. Se o arquivo for válido, isso preencherá automaticamente todos os campos de resumo no console. Ou você pode copiar e colar manualmente os seguintes valores diretamente do seu arquivo de resumo:

   - Resumo: o valor `digest` do seu arquivo de resumo.
   - Endereço dica de resumo: o valor `digestTipAddress` do seu arquivo de resumo.
8. Revise os parâmetros de entrada do documento e do resumo e escolha Verificar.

   O console automatiza duas etapas para você:

   a. Solicite um formulário de prova QLDB para o documento especificado.
   b. Use a prova retornada por QLDB para ligar para um clienteAPI, que verifica a revisão do documento em relação ao resumo fornecido.

   O console exibe os resultados da sua solicitação no cartão Resultados da verificação. Para ter mais informações, consulte Resultados da verificação.
9. Para testar a lógica de verificação, repita as etapas 6 a 8 em Para verificar uma revisão do documento, mas altere um único caractere na string de entrada Digest. Isso deve fazer com que sua solicitação de verificação falhe com uma mensagem de erro apropriada.

Se você não precisar mais usar o ledger `vehicle-registration`, prossiga para Etapa 7 (opcional): Limpar os recursos.

Etapa 7 (opcional): Limpar os recursos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode continuar usando o ledger `vehicle-registration`. No entanto, se não precisar mais dele, deverá excluí-lo.

Se a proteção contra exclusão estiver ativada para seu livro contábil, você deverá primeiro desativá-la antes de excluir o livro contábil usando o QLDBAPI, AWS Command Line Interface (AWS CLI) ou o console. QLDB

Para excluir o ledger

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Se a proteção contra exclusão estiver habilitada para este ledger, você deverá desabilitá-la primeiro.

   Na lista de ledgers, selecione `vehicle-registration` e escolha Editar ledger.
4. Na página Editar ledger, desative a Proteção contra exclusão e escolha Confirmar alterações.
5. Na lista de ledgers, selecione `vehicle-registration` novamente e escolha Excluir.
6. Confirme isso inserindo **`delete vehicle-registration`** no campo fornecido.

Para saber mais sobre como trabalhar com livros contábeis emQLDB, consulte. Começando com a AmazonQLDB: próximas etapas

Começando com a AmazonQLDB: próximas etapas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Para obter mais informações sobre o uso da AmazonQLDB, consulte os seguintes tópicos:

- Trabalhando com dados e histórico na Amazon QLDB
- Começando com o QLDB driver da Amazon
- Modelo de QLDB concorrência da Amazon
- Exportação de dados de periódicos da Amazon QLDB
- Streaming de dados de diários da Amazon QLDB
- Verificação de dados na Amazon QLDB
- Referência do Amazon QLDB PartiQL

Começando com o QLDB driver da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este capítulo contém tutoriais práticos para ajudar você a aprender sobre o desenvolvimento com a Amazon QLDB usando o driver. QLDB O driver é construído em cima do AWS SDK, que suporta a interação com QLDBAPIo.

QLDBabstração da sessão

O driver fornece uma camada de abstração de alto nível acima dos dados API transacionais (Sessão). QLDB Ele simplifica o processo de execução de instruções partiQL em dados contábeis gerenciando chamadas. SendCommandAPI Essas API chamadas exigem vários parâmetros que o driver manipula para você, incluindo o gerenciamento de sessões, transações e política de repetição em caso de erros. O motorista também tem otimizações de desempenho e aplica as melhores práticas para interagir com. QLDB

> **Note**
>
> Para interagir com as API operações de gerenciamento de recursos listadas na QLDBAPIreferência da Amazon, você usa o AWS SDK diretamente em vez do driver. Você usa o gerenciamento API somente para gerenciar recursos contábeis e para operações de dados não transacionais, como exportação, streaming e verificação de dados.

Suporte ao Amazon Ion

Além disso, o driver usa as bibliotecas Amazon Ion para fornecer suporte para lidar com dados de íons ao executar transações. Essas bibliotecas também cuidam do cálculo do hash dos valores de Ion. QLDBrequer esses hashes de íons para verificar a integridade das solicitações de transação de dados.

Terminologia do driver

Essa ferramenta é chamada de driver porque é comparável a outros drivers de banco de dados que fornecem interfaces amigáveis ao desenvolvedor. Esses drivers também encapsulam a lógica que converte um conjunto padrão de comandos e funções em chamadas específicas que são exigidas pelo nível inferior do serviço. API

O driver é de código aberto GitHub e está disponível para as seguintes linguagens de programação:

- driver Java
- . NETmotorista
- Driver Go
- driver Node.js
- Driver Python

Para obter informações gerais sobre drivers para todas as linguagens de programação suportadas e tutoriais adicionais, consulte os tópicos a seguir:

- Gerenciamento da sessão com o driver
- Recomendações de driver
- Política de novas tentativas de driver
- Erros comuns
- Tutorial de aplicativo de exemplo
- Como trabalhar com o Amazon Ion
- Obter estatísticas de instruções PartiQL

QLDBDriver Amazon para Java

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Para trabalhar com dados em seu livro contábil, você pode se conectar à Amazon a QLDB partir do seu aplicativo Java usando um driver AWS fornecido. Os tópicos a seguir descrevem como começar a usar o QLDB driver para Java.

Tópicos

- Recursos para driver
- Pré-requisitos
- Como credenciar suas credenciais AWS e região padrão
- Instalação
- QLDBDriver Amazon para Java — Tutorial de início rápido
- QLDBDriver Amazon para Java — Referência do livro de receitas

Recursos para driver

Para obter mais informações sobre a funcionalidade suportada pelo driver Java, consulte os recursos a seguir:

- APIreferência: 2.x, 1.x
- Código-fonte do driver (GitHub)
- Exemplo de código-fonte do aplicativo (GitHub)
- Estrutura do carregador de livros contábeis () GitHub
- Exemplos de código do Amazon Ion

Pré-requisitos

Antes de começar a usar o QLDB driver para Java, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando a Amazon QLDB. Essa transmissão inclui o seguinte:
    1. Inscreva-se em AWS.
    2. Crie um usuário com as QLDB permissões apropriadas.
    3. Conceda acesso programático para desenvolvimento.
2. Configure um ambiente de desenvolvimento Java baixando e instalando o seguinte:

   1. Java SE Development Kit 8, como o Amazon Corretto 8.
   2. (Opcional) Ambiente de desenvolvimento integrado Java (IDE) de sua escolha, como Eclipse ou IntelliJ.
3. Configure seu ambiente de desenvolvimento para AWS SDK for Java sempreComo credenciar suas credenciais AWS e região padrão.

Em seguida, você pode baixar o aplicativo completo de amostra do tutorial ou instalar somente o driver em um projeto Java e executar exemplos de códigos curtos.

- Para instalar o QLDB driver e o AWS SDK for Java em um projeto existente, vá paraInstalação.
- Para configurar um projeto e executar exemplos de códigos curtos que demonstram transações básicas de dados em um ledger, consulte o Tutorial de início rápido.
- Para executar exemplos mais detalhados de API operações de dados e gerenciamento no aplicativo de amostra completo do tutorial, consulte o. Tutorial de Java

Como credenciar suas credenciais AWS e região padrão

O QLDB driver e o subjacente AWS SDK for Javaexigem que você forneça AWS credenciais ao seu aplicativo em tempo de execução. Os exemplos de código neste guia pressupõem que você esteja usando um arquivo de credenciais AWS , conforme descrito em Configurar credenciais e região padrão no Guia do desenvolvedor do AWS SDK for Java 2.x .

Como parte dessas etapas, você também deve definir seu padrão Região da AWS para determinar seu QLDB endpoint padrão. Os exemplos de código se conectam ao QLDB seu padrão Região da AWS. Para obter uma lista completa das regiões onde QLDB está disponível, consulte QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Veja a seguir um exemplo de um arquivo de AWS credenciais chamado`~/.aws/credentials`, em que o caractere tilde (~) representa seu diretório inicial.

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

Substitua seus próprios valores AWS de credenciais pelos valores *`your_access_key_id`* e *`your_secret_access_key`*.

Instalação

QLDBsuporta as seguintes versões do driver Java e suas AWS SDK dependências.

Versão do driver	AWS SDK	Status	Data da versão mais recente
1.x	AWS SDK for Java 1.x	Lançamento de produção	20 de março de 2020
2.x	AWS SDK for Java 2.x	Lançamento de produção	04 de junho de 2021

Para instalar o QLDB driver, recomendamos usar um sistema de gerenciamento de dependências, como o Gradle ou o Maven. Por exemplo, adicione o artefato a seguir como uma dependência no seu projeto Java.

2.x

Gradle

Adicione essa dependência ao seu arquivo `build.gradle` de configuração.

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

Maven

Adicione essa dependência ao seu arquivo `pom.xml` de configuração.

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

Esse artefato inclui automaticamente o módulo AWS SDK for Java 2.x principal, as bibliotecas Amazon Ion e outras dependências necessárias.

1.x

Gradle

Adicione essa dependência ao seu arquivo `build.gradle` de configuração.

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

Maven

Adicione essa dependência ao seu arquivo `pom.xml` de configuração.

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

Esse artefato inclui automaticamente o módulo AWS SDK for Java principal, as bibliotecas Amazon Ion e outras dependências necessárias.

> **Important**
>
> Namespace Amazon Ion: ao importar classes Amazon Ion em seu aplicativo, você deve usar o pacote que está sob o namespace`com.amazon.ion`. AWS SDK for Java Depende de outro pacote Ion no namespace`software.amazon.ion`, mas esse é um pacote legado que não é compatível com o QLDB driver.

Para exemplos de código curto de como executar transações básicas de dados em um ledger, consulte o Referência de Cookbook.

Outras bibliotecas opcionais

Como opção, adicione também as seguintes bibliotecas úteis ao seu projeto. Esses artefatos são dependências obrigatórias no aplicativo Tutorial de Java de amostra.

1. aws-java-sdk-qldb— O QLDB módulo do AWS SDK for Java. A versão mínima QLDB suportada é`1.11.785`.

   Use esse módulo em seu aplicativo para interagir diretamente com as API operações de gerenciamento listadas noQLDBAPIReferência da Amazon.

2. jackson-dataformat-ion— Módulo XML de formato de dados Jackson da Faster para Ion. O aplicativo de amostra requer uma versão `2.10.0` ou posterior.

Gradle

Adicione essas dependências ao seu arquivo de configuração `build.gradle`.

```
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

Adicione essas dependências ao seu arquivo de configuração `pom.xml`.

```
<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>
```

QLDBDriver Amazon para Java — Tutorial de início rápido

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você aprende a configurar um aplicativo simples usando a versão mais recente do QLDB driver da Amazon para Java. Este guia inclui etapas para instalar o driver e exemplos de códigos curtos das operações básicas de criação, leitura, atualização e exclusão (CRUD). Para obter exemplos mais detalhados que demonstram essas operações em um aplicativo de amostra completo, consulte o Tutorial de Java.

Tópicos

- Pré-requisitos
- Etapa 1: configurar o projeto do
- Etapa 2: Inicializar o driver
- Etapa 3: Crie uma tabela e um índice
- Etapa 4: Inserir um documento
- Etapa 5: consultar o documento
- Etapa 6: Atualize o documento
- Executar o aplicativo completo

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Complete a Pré-requisitos para o driver Java, caso ainda não o tenha feito. Isso inclui se inscrever AWS, conceder acesso programático para desenvolvimento e instalar um ambiente de desenvolvimento integrado Java ()IDE.
2. Crie um ledger chamado `quick-start`.

   Para saber como criar um ledger, consulte Operações básicas para livros contábeis da Amazon QLDB ou Etapa 1: criar um novo ledger em Conceitos básicos do console.

Etapa 1: configurar o projeto do

Primeiro, configure seu projeto Java. Recomendamos usar o sistema de gerenciamento de dependências Maven para este tutorial.

> **Note**
>
> Se você usa um IDE que tenha recursos para automatizar essas etapas de configuração, você pode pular para. Etapa 2: Inicializar o driver

1. Crie uma pasta para o seu aplicativo.

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

2. Digite o comando a seguir para Inicializar o projeto a partir de um modelo Maven. Substituir *project-package*, *project-name* e *maven-template* com seus próprios valores, conforme apropriado.

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

   Para *maven-template*, você pode usar o modelo básico do Maven: `maven-archetype-quickstart`

3. Para adicionar o QLDBdriver para Java como uma dependência do projeto, navegue até o `pom.xml` arquivo recém-criado e adicione o seguinte artefato.

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

Esse artefato inclui automaticamente o módulo AWS SDK for Java 2.x principal, as bibliotecas Amazon Ion e outras dependências necessárias. Seu arquivo `pom.xml` deverá ser semelhante ao seguinte:

```
<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. Abra o arquivo `App.java`.

   Em seguida, adicione incrementalmente os exemplos de código nas etapas a seguir para experimentar algumas CRUD operações básicas. Ou você pode pular o step-by-step tutorial e, em vez disso, executar o aplicativo completo.

Etapa 2: Inicializar o driver

Inicialize uma instância do driver que se conecta ao ledger chamado `quick-start`. Adicione o seguinte código ao arquivo `App.java`.

```
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();
    }
}
```

Etapa 3: Crie uma tabela e um índice

O exemplo de código a seguir mostra como executar as instruções `CREATE TABLE` e `CREATE INDEX`.

No método `main`, adicione o código a seguir que cria uma tabela chamada `People` e um índice para o campo `lastName` nessa tabela. Os índices são necessários para otimizar o desempenho da consulta e ajudar a limitar as exceções otimistas de conflitos do controle de simultaneidade (OCC).

```
// 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)");
});
```

Etapa 4: Inserir um documento

O exemplo de código a seguir mostra como executar uma instrução `INSERT`. QLDBsuporta a linguagem de consulta partiQL (SQLcompatível) e o formato de dados Amazon Ion (superconjunto de). JSON

Adicione o código a seguir que insere um documento na tabela `People`.

```
// 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);
});
```

Este exemplo usa um ponto de interrogação (?) como um marcador variável para passar as informações do documento para a instrução. Ao usar espaços reservados, você deve passar um valor do tipo `IonValue`.

> **Tip**
>
> Para inserir vários documentos usando uma única INSERT instrução, você pode passar um parâmetro do tipo IonList(convertido explicitamente como um`IonValue`) para a instrução da seguinte maneira.
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```
>
> Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma `IonList`. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Etapa 5: consultar o documento

O exemplo de código a seguir mostra como executar uma instrução `SELECT`.

Adicione o código a seguir que insere um documento da tabela `People`.

```
// 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
});
```

Etapa 6: Atualize o documento

O exemplo de código a seguir mostra como executar uma instrução `UPDATE`.

1. Adicione o código a seguir que insere um documento na tabela `People`, atualizando `age` para `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. Consulte a tabela novamente para ver o valor atualizado.

   ```
   // 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. Use o Maven ou o seu IDE para compilar e executar o `App.java` arquivo.

Executar o aplicativo completo

O exemplo de código a seguir é a versão completa do aplicativo `App.java`. Em vez de executar as etapas anteriores individualmente, você também pode copiar e executar esse exemplo de código do início ao fim. Este aplicativo demonstra algumas CRUD operações básicas no livro contábil chamado. `quick-start`

> **Note**
>
> Antes de executar esse código, verifique se você ainda não tem uma tabela ativa chamada `People` no ledger `quick-start`.
> Na primeira linha, substitua *`project-package`* com o `groupId` valor que você usou para o comando Maven emEtapa 1: configurar o projeto do .

```
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
        });
    }
}
```

Use o Maven ou o seu IDE para compilar e executar o `App.java` arquivo.

QLDBDriver Amazon para Java — Referência do livro de receitas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este guia de referência mostra casos de uso comuns do QLDB driver da Amazon para Java. Ele fornece exemplos de código Java que demonstram como usar o driver para executar operações básicas de criação, leitura, atualização e exclusão (CRUD). Também inclui exemplos de código para processamento de dados do Amazon Ion. Além disso, este guia destaca as práticas recomendadas para tornar as transações idempotentes e implantar restrições de exclusividade.

> **Note**
>
> Quando aplicável, alguns casos de uso têm exemplos de código diferentes para cada versão principal compatível do QLDB driver para Java.

Sumário

- Importação do driver
- Instanciação do driver
- CRUDoperações
  - Criar tabelas
  - Criar índices
  - Ler documentos
  - Inserir documentos
    - Como inserir vários documentos em uma instrução

- Como atualizar documentos
- Como excluir documentos
- Executando várias instruções em uma transação
- Lógica de novas tentativas
- Implementação de restrições de exclusividade
- Como trabalhar com o Amazon Ion
  - Importando os pacotes Ion
  - Inicializando Ion
  - Criação de objetos do Ion
  - Leitura de objetos do Ion

Importação do driver

O exemplo de código a seguir importa o driver, o cliente da QLDB sessão, os pacotes Amazon Ion e outras dependências relacionadas.

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;
```

Instanciação do driver

O exemplo de código a seguir cria uma instância de driver que se conecta a um nome do ledger especificado e usa a lógica de repetição especificada com um limite de repetição personalizado.

> **Note**
>
> Este exemplo também instancia um objeto do sistema Amazon Ion (`IonSystem`). Você precisa desse objeto para processar dados Ion ao executar algumas operações de dados nesta referência. Para saber mais, consulte Como trabalhar com o Amazon Ion.

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();
```

CRUDoperações

QLDBexecuta operações de criação, leitura, atualização e exclusão (CRUD) como parte de uma transação.

> **Warning**
>
> Como prática recomendada, torne suas transações de gravação estritamente idempotentes.

Tornar as transações idempotentes

Recomendamos que você torne as transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de novas tentativas. Uma transação é idempotente se puder ser executada várias vezes e produzir resultados idênticos a cada vez.

Por exemplo, considere uma transação que insere um documento em uma tabela chamada `Person`. A transação deve primeiro verificar se o documento já existe ou não na tabela. Sem essa verificação, a tabela pode acabar com documentos duplicados.

Suponha que a transação seja confirmada QLDB com sucesso no lado do servidor, mas o cliente expire enquanto espera por uma resposta. Se a transação não for idempotente, o mesmo documento poderá ser inserido mais de uma vez no caso de uma nova tentativa.

Usar índices para evitar varreduras completas da tabela

Também recomendamos executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Sem essa pesquisa indexada, é QLDB necessário fazer uma varredura da tabela, o que pode levar a tempos limite de transação ou conflitos otimistas de controle de simultaneidade (). OCC

Para obter mais informações sobre o OCC, consulte Modelo de QLDB concorrência da Amazon.

Transações criadas implicitamente

O método QldbDriver.execute aceita uma função lambda que recebe uma instância do Executor, que você pode usar para executar instruções. A instância `Executor` envolve uma transação criada implicitamente.

Você pode executar instruções na função do Lambda usando o método `Executor.execute`. O driver confirma implicitamente a transação quando a função do Lambda retorna.

As seções a seguir mostram como executar CRUD operações básicas, especificar a lógica de repetição personalizada e implementar restrições de exclusividade.

> **Note**
>
> Quando aplicável, essas seções fornecem exemplos de código de processamento de dados do Amazon Ion usando a biblioteca Ion integrada e a biblioteca Jackson Ion mapper. Para saber mais, consulte Como trabalhar com o Amazon Ion.

Sumário

- Criar tabelas
- Criar índices
- Ler documentos
- Inserir documentos
  - Como inserir vários documentos em uma instrução
- Como atualizar documentos
- Como excluir documentos
- Executando várias instruções em uma transação
- Lógica de novas tentativas
- Implementação de restrições de exclusividade

Criar tabelas

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

Criar índices

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

Ler documentos

```
// 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
});
```

Como usar parâmetros de consulta

O exemplo de código a seguir usa um parâmetro de consulta do tipo 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
});
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta.

```
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
});
```

O exemplo de código a seguir usa uma lista de parâmetros de consulta.

```
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
```

```
});
```

Usar o mapeador Jackson

```
// 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();
    }
});
```

Como usar parâmetros de consulta

O exemplo de código a seguir usa um parâmetro de consulta do tipo 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();
    }
});
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta.

```
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();
    }
});
```

O exemplo de código a seguir usa uma lista de parâmetros de consulta.

```
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**
>
> Quando você executa uma consulta sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as consultas podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserir documentos

Os exemplos de código a seguir inserem os tipos de dados Ion.

```
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);
    }
});
```

Usar o mapeador Jackson

Os exemplos de código a seguir inserem os tipos de dados Ion.

```
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();
    }
});
```

Essa transação insere um documento na tabela `Person`. Antes de inserir, ele primeiro verifica se o documento já existe na tabela. Essa verificação torna a transação idempotente por natureza. Mesmo que você execute essa transação várias vezes, ela não causará efeitos colaterais indesejados.

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como inserir vários documentos em uma instrução

Para inserir vários documentos usando uma única INSERT instrução, você pode passar um parâmetro do tipo IonList(convertido explicitamente como um`IonValue`) para a instrução da seguinte maneira.

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

Você não coloca o marcador variável (`?`) entre colchetes angulares duplos (`<<...>>`) ao passar uma `IonList`. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Por que o elenco explícito é necessário?

O TransactionExecutormétodo.execute está sobrecarregado. Ele aceita um número variável de `IonValue` argumentos (varargs) ou um único argumento `List<IonValue>`. Em ion-java, `IonList` é implementado como um `List<IonValue>`.

O Java usa como padrão a implantação do método mais específico quando você chama um método sobrecarregado. Nesse caso, quando você passa um parâmetro `IonList`, o padrão é o método que usa a `List<IonValue>`. Quando invocada, essa implementação de método passa os elementos `IonValue` da lista como valores distintos. Portanto, para invocar o método varargs em vez disso, você deve converter explicitamente um parâmetro `IonList` em `IonValue`.

Como atualizar documentos

```
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);
});
```

Usar o mapeador Jackson

```
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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como excluir documentos

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

Usar o mapeador Jackson

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

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Executando várias instruções em uma transação

```
// 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;
    });
}
```

Lógica de novas tentativas

O `execute` método do driver tem um mecanismo de repetição integrado que repete a transação se ocorrer uma exceção que pode ser repetida (como tempos limite ou conflitos). OCC

2.x

O número máximo de tentativas de repetição e a estratégia de recuo são configuráveis.

O limite padrão de repetição é`4`, e a estratégia de recuo padrão é. DefaultQldbTransactionBackoffStrategy Você pode definir a configuração de repetição por instância de driver e também por transação usando uma instância de RetryPolicy.

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância do driver.

```
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());
    }
}
```

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância particular. Essa configuração para `execute` substitui a lógica de repetição definida para a instância do driver.

```
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

O número máximo de tentativas de repetição pode ser configurado. Você pode configurar o limite de novas tentativas definindo a propriedade `retryLimit` ao inicializar `PooledQldbDriver`.

O limite padrão de novas tentativas é `4`.

Implementação de restrições de exclusividade

QLDBnão oferece suporte a índices exclusivos, mas você pode implementar esse comportamento em seu aplicativo.

Suponha que você queira implementar uma restrição de exclusividade no campo `GovId` da tabela `Person`. Para fazer isso, você pode escrever uma transação que faça o seguinte:

1. Afirme que a tabela não tem documentos existentes com um `GovId` especificado.
2. Insira o documento se a afirmação for aprovada.

Se uma transação concorrente passar simultaneamente pela declaração, somente uma das transações será confirmada com sucesso. A outra transação falhará com uma exceção de OCC conflito.

O exemplo de código a seguir mostra como implementar essa lógica de restrição de exclusividade.

```
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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como trabalhar com o Amazon Ion

Há várias maneiras de processar dados do Amazon Ion noQLDB. Você pode usar métodos integrados da biblioteca Ion para criar e modificar documentos com flexibilidade, conforme necessário. Ou você pode usar o módulo de formato XML de dados Jackson da Faster para Ion para mapear documentos Ion para modelos Java object (POJO) simples e antigos.

As seções a seguir fornecem exemplos de código de processamento de dados Ion usando ambas as técnicas.

Sumário

- Importando os pacotes Ion
- Inicializando Ion
- Criação de objetos do Ion
- Leitura de objetos do Ion

Importando os pacotes Ion

Adicione o artefato ion-java como uma dependência em seu projeto Java.

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>
```

Importe os seguintes pacotes de Ion.

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

Usar o mapeador Jackson

Adicione o artefato jackson-dataformat-ioncomo uma dependência em seu projeto Java. QLDBrequer uma versão `2.10.0` ou posterior.

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>
```

Importe os seguintes pacotes de Ion.

```
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;
```

Inicializando Ion

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

Usar o mapeador Jackson

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

Criação de objetos do Ion

O exemplo de código a seguir cria um objeto Ion usando a interface `IonStruct` e seus métodos integrados.

```
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
```

Usar o mapeador Jackson

Suponha que você tenha uma classe JSON de modelo mapeada com o nome `Person` a seguir.

```
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;
    }
}
```

O exemplo de código a seguir cria um objeto `IonStruct` a partir de uma instância de `Person`.

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

Leitura de objetos do Ion

O exemplo de código a seguir imprime cada campo da instância `ionStruct`.

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

Usar o mapeador Jackson

O exemplo de código a seguir lê o objeto e mapas de `IonStruct` a partir de uma instância de `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
```

Para obter mais informações sobre como trabalhar com o Ion, consulte a documentação do Amazon Ion em GitHub. Para obter mais exemplos de código de como trabalhar com o Ion inQLDB, consulteTrabalhando com os tipos de dados Amazon Ion na Amazon QLDB.

QLDBDriver da Amazon para. NET

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Para trabalhar com dados em seu livro contábil, você pode se conectar à Amazon QLDB usando seu Microsoft. NETaplicativo usando um driver AWS fornecido. Os alvos do motorista. NETPadrão 2.0. Mais especificamente, ele suporta. NETCore (LTS) 2.1+ e. NETEstrutura 4.5.2+. Para obter informações sobre compatibilidade, consulte. NETPadrão no site do Microsoft Docs.

É altamente recomendável usar o mapeador de objetos Ion para ignorar completamente a necessidade de conversão manual entre tipos Amazon Ion e tipos nativos de C#.

Os tópicos a seguir descrevem como começar a usar o QLDB driver do. NET.

Tópicos

- Recursos para driver
- Pré-requisitos
- Instalação
- QLDBDriver da Amazon para. NET— Tutorial de início rápido
- QLDBDriver da Amazon para. NET— Referência do livro de receitas

Recursos para driver

Para obter mais informações sobre a funcionalidade suportada pelo. NETmotorista, consulte os seguintes recursos:

- APIreferência
- Código-fonte do driver (GitHub)
- Exemplo de código-fonte do aplicativo (GitHub)
- Cookbook Amazon Ion
- Mapeador de objetos de íons () GitHub

Pré-requisitos

Antes de começar com o QLDB driver do. NET, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando a Amazon QLDB. Essa transmissão inclui o seguinte:

    1. Inscreva-se em AWS.

   2. Crie um usuário com as QLDB permissões apropriadas.
   3. Conceda acesso programático para desenvolvimento.
2. Baixe e instale o. NETCore SDK versão 2.1 ou posterior da Microsoft. NETsite de downloads.
3. (Opcional) Instale um ambiente de desenvolvimento integrado (IDE) de sua escolha, como Visual Studio, Visual Studio para Mac ou Visual Studio Code. Você pode baixá-los do site do Microsoft Visual Studio.
4. Configure seu ambiente de desenvolvimento para AWS SDK for .NET.
   1. Configure suas AWS credenciais. Recomendamos criar um arquivo de credenciais compartilhadas.

      Para obter instruções, consulte Como configurar credenciais AWS usando um arquivo de credenciais no Guia do desenvolvedor do AWS SDK for .NET .
   2. Defina sua Região da AWS padrão. Para saber como, consulte seleção Região da AWS.

      Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Em seguida, você pode configurar um aplicativo de amostra básico e executar exemplos de códigos curtos, ou você pode instalar o driver em um existente. NETprojeto.

- Para instalar o QLDB driver e o AWS SDK for .NET em um projeto existente, vá paraInstalação.
- Para configurar um projeto e executar exemplos de códigos curtos que demonstram transações básicas de dados em um ledger, consulte o Tutorial de início rápido.

Instalação

Use o gerenciador de NuGet pacotes para instalar o QLDB driver para. NET. Recomendamos usar o Visual Studio ou um IDE de sua escolha para adicionar dependências do projeto. O nome do pacote do driver é Amazon. QLDB.Motorista.

Por exemplo, no Visual Studio, abra o NuGet Package Manager Console no menu Tools. Então, digite o seguinte comando no `PM>` prompt:

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

A instalação do driver também instala suas dependências, incluindo os pacotes Amazon Ion e AWS SDK for .NET Amazon.

Instale o mapper de objetos Ion

Versão 1.3.0 do QLDB driver para. NETintroduz suporte para aceitar e retornar tipos de dados C# nativos sem a necessidade de trabalhar com o Amazon Ion. Para usar esse atributo, adicione o seguinte pacote ao seu projeto.

- Amazon. QLDB.Driver.Serialization — Uma biblioteca que pode mapear valores de íons para CLRobjetos antigos simples (POCO) em C# e vice-versa. Esse mapper de objetos Ion permite que seu aplicativo interaja diretamente com os tipos de dados C# nativos sem a necessidade de trabalhar com o Ion. Para obter um breve guia sobre como usar essa biblioteca, consulte o SERIALIZATIONarquivo.md no GitHub repositório`awslabs/amazon-qldb-driver-dotnet`.

Para instalar esse pacote, insira o seguinte comando:

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

Para exemplos de código curto de como executar transações básicas de dados em um ledger, consulte o Referência de Cookbook.

QLDBDriver da Amazon para. NET— Tutorial de início rápido

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você aprende como configurar um aplicativo simples usando o QLDB driver da Amazon para. NET. Este guia inclui etapas para instalar o driver e exemplos de códigos curtos das operações básicas de criação, leitura, atualização e exclusão (CRUD).

Tópicos

- Pré-requisitos
- Etapa 1: configurar o projeto do

- Etapa 2: Inicializar o driver
- Etapa 3: Crie uma tabela e um índice
- Etapa 4: Inserir um documento
- Etapa 5: consultar o documento
- Etapa 6: Atualize o documento
- Executar o aplicativo completo

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Complete o Pré-requisitos para o. NETmotorista, se você ainda não o fez. Isso inclui se inscrever AWS, conceder acesso programático para desenvolvimento e instalar o. NETNúcleoSDK.
2. Crie um ledger chamado `quick-start`.

   Para saber como criar um ledger, consulte Operações básicas para livros contábeis da Amazon QLDB ou Etapa 1: criar um novo ledger em Conceitos básicos do console.

Etapa 1: configurar o projeto do

Primeiro, configure seu. NETprojeto.

1. Para criar e executar um aplicativo modelo, digite os seguintes `dotnet` comandos em um terminal PowerShell, como bash ou prompt de comando.

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

   Esse modelo cria uma pasta chamada`Amazon.QLDB.QuickStartGuide`. Nessa pasta, ele cria um projeto com o mesmo nome e um arquivo chamado `Program.cs`. O programa contém código que exibe a saída `Hello World!`.
2. Use o gerenciador de NuGet pacotes para instalar o QLDB driver para. NET. Recomendamos usar o Visual Studio ou um IDE de sua escolha para adicionar dependências ao seu projeto. O nome do pacote do driver é Amazon. QLDB.Motorista.
   - Por exemplo, no Visual Studio, abra o NuGet Package Manager Console no menu Tools. Então, digite o seguinte comando no `PM>` prompt:

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

- Ou, é possível inserir os comandos a seguir no terminal.

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

A instalação do driver também instala suas dependências, incluindo os pacotes AWS SDK for .NET e Amazon Ion.

3. Instale a biblioteca de serialização do driver.

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

4. Abra o arquivo `Program.cs`.

   Em seguida, adicione incrementalmente os exemplos de código nas etapas a seguir para experimentar algumas CRUD operações básicas. Ou você pode pular o step-by-step tutorial e, em vez disso, executar o aplicativo completo.

> **Note**
>
> - Escolha entre síncrono e assíncrono APIs — O driver fornece síncrono e assíncrono. APIs Para aplicativos de alta demanda que lidam com várias solicitações sem bloqueio, recomendamos usar o APIs assíncrono para melhorar o desempenho. O driver oferece síncrono APIs como uma conveniência adicional para bases de código existentes que são escritas de forma síncrona.
>
> Este tutorial inclui exemplos de código síncrono e assíncrono. Para obter mais informações sobre oAPIs, consulte as IAsyncQldbDriverinterfaces IQldbDrivere na API documentação.
>
> - Processamento de dados Amazon Ion: este tutorial fornece exemplos de código de processamento de dados do Amazon Ion usando o mapeador de objetos Ion por padrão. QLDBintroduziu o mapeador de objetos Ion na versão 1.3.0 do. NETmotorista. Onde aplicável, este tópico também fornece exemplos de código usando a biblioteca Ion padrão como alternativa. Para saber mais, consulte Como trabalhar com o Amazon Ion.

Etapa 2: Inicializar o driver

Inicialize uma instância do driver que se conecta ao ledger chamado `quick-start`. Adicione o seguinte código ao arquivo `Program.cs`.

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();
        }
    }
}
```

Usar a biblioteca ION

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();
        }
    }
}
```

Etapa 3: Crie uma tabela e um índice

Para o restante deste tutorial até a Etapa 6, você precisa acrescentar os exemplos de código a seguir ao exemplo de código anterior.

Os exemplos de código a seguir mostram como executar as instruções `CREATE TABLE` e `CREATE INDEX`. Adicione o código a seguir que cria uma tabela chamada `Person` e um índice para o campo `firstName` nessa tabela. Os índices são necessários para otimizar o desempenho da consulta e ajudar a limitar as exceções otimistas de conflitos do controle de simultaneidade (OCC).

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)");
});
```

Etapa 4: Inserir um documento

O exemplo de código a seguir mostra como executar uma instrução `INSERT`. QLDBsuporta a linguagem de consulta partiQL (SQLcompatível) e o formato de dados Amazon Ion (superconjunto de). JSON

Adicione o código a seguir que insere um documento na tabela `Person`.

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);
});
```

Usar a biblioteca ION

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**
>
> Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista Ion para a instrução da seguinte maneira.

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

Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista Ion. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Etapa 5: consultar o documento

O exemplo de código a seguir mostra como executar uma instrução `SELECT`.

Adicione o código a seguir que insere um documento da tabela `Person`.

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
}
```

Usar a biblioteca ION

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);
}
```

Este exemplo usa um ponto de interrogação (?) como um marcador variável para passar as informações do documento para a instrução. Ao usar espaços reservados, você deve passar um valor do tipo `IonValue`.

Etapa 6: Atualize o documento

O exemplo de código a seguir mostra como executar uma instrução `UPDATE`.

1. Adicione o código a seguir que insere um documento na tabela `Person`, atualizando `age` para 42.

   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. Consulte a tabela novamente para ver o valor atualizado.

   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. Para executar o aplicativo, insira o comando a seguir do diretório `Amazon.QLDB.QuickStartGuide` do seu projeto.

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

Usar a biblioteca ION

1. Adicione o código a seguir que insere um documento na tabela `Person`, atualizando `age` para 42.

   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. Consulte a tabela novamente para ver o valor atualizado.

   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. Para executar o aplicativo, insira o comando a seguir do diretório `Amazon.QLDB.QuickStartGuide` do seu projeto.

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

Executar o aplicativo completo

O exemplo de código a seguir é a versão completa do aplicativo `Program.cs`. Em vez de executar as etapas anteriores individualmente, você também pode copiar e executar esse exemplo de código do início ao fim. Este aplicativo demonstra algumas CRUD operações básicas no livro contábil chamado. `quick-start`

> **Note**
>
> Antes de executar esse código, verifique se você ainda não tem uma tabela ativa chamada `Person` no ledger `quick-start`.

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
            }

        }
    }
}
```

Usar a biblioteca ION

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);
            }
        }
    }
}
```

Para executar o aplicativo, insira o comando a seguir do diretório principal do seu diretório de projeto `Amazon.QLDB.QuickStartGuide`.

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

QLDBDriver da Amazon para. NET— Referência do livro de receitas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este guia de referência mostra casos de uso comuns do QLDB driver da Amazon para. NET. Ele fornece exemplos de código C# que demonstram como usar o driver para executar operações básicas de criação, leitura, atualização e exclusão (CRUD). Também inclui exemplos de código para processamento de dados do Amazon Ion. Além disso, este guia destaca as práticas recomendadas para tornar as transações idempotentes e implantar restrições de exclusividade.

> **Note**
>
> Este tópico fornece exemplos de código de processamento de dados do Amazon Ion usando o mapeador de objetos Ion por padrão. QLDBintroduziu o mapeador de objetos Ion na versão 1.3.0 do. NETmotorista. Onde aplicável, este tópico também fornece exemplos de código usando a biblioteca Ion padrão como alternativa. Para saber mais, consulte Como trabalhar com o Amazon Ion.

Sumário

- Importação do driver
- Instanciação do driver
- CRUDoperações
  - Criar tabelas
  - Criar índices
  - Ler documentos
    - Usar parâmetros de consulta
  - Inserir documentos
    - Como inserir vários documentos em uma instrução
  - Como atualizar documentos

- Como excluir documentos
- Como executar várias instruções em uma transação
- Lógica de novas tentativas
- Implementação de restrições de exclusividade
- Como trabalhar com o Amazon Ion
  - Importar o módulo Ion
  - Criação de tipos de Ion
  - Obter um despejo de binário Ion
  - Obter um despejo de texto Ion

Importação do driver

O exemplo de código a seguir importa o driver.

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

Usar a biblioteca ION

```
using Amazon.QLDB.Driver;
using Amazon.IonDotnet.Builders;
```

Instanciação do driver

O exemplo de código a seguir cria uma instância do driver que se conecta a um nome de ledger especificado usando as configurações padrão.

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();
```

Usar a biblioteca ION

Async

```
IAsyncQldbDriver driver = AsyncQldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

Sync

```
IQldbDriver driver = QldbDriver.Builder().WithLedger("vehicle-
registration").Build();
```

CRUDoperações

QLDBexecuta operações de criação, leitura, atualização e exclusão (CRUD) como parte de uma transação.

> **Warning**
>
> Como prática recomendada, torne suas transações de gravação estritamente idempotentes.

Tornar as transações idempotentes

Recomendamos que você torne as transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de novas tentativas. Uma transação é idempotente se puder ser executada várias vezes e produzir resultados idênticos a cada vez.

Por exemplo, considere uma transação que insere um documento em uma tabela chamada `Person`. A transação deve primeiro verificar se o documento já existe ou não na tabela. Sem essa verificação, a tabela pode acabar com documentos duplicados.

Suponha que a transação seja confirmada QLDB com sucesso no lado do servidor, mas o cliente expire enquanto espera por uma resposta. Se a transação não for idempotente, o mesmo documento poderá ser inserido mais de uma vez no caso de uma nova tentativa.

Usar índices para evitar varreduras completas da tabela

Também recomendamos executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Sem essa pesquisa indexada, é QLDB necessário fazer uma varredura da tabela, o que pode levar a tempos limite de transação ou conflitos otimistas de controle de simultaneidade (). OCC

Para obter mais informações sobre o OCC, consulte Modelo de QLDB concorrência da Amazon.

Transações criadas implicitamente

A Amazônia. QLDB.Motorista. IQldbDriverO método.Execute aceita uma função lambda que recebe uma instância da Amazon. QLDB.Motorista. TransactionExecutor, que você pode usar para executar instruções. A instância de `TransactionExecutor` envolve uma transação criada implicitamente.

Você pode executar instruções na função do Lambda usando o método `Execute` do executor da transação. O driver confirma implicitamente a transação quando a função do Lambda retorna.

As seções a seguir mostram como executar CRUD operações básicas, especificar a lógica de repetição personalizada e implementar restrições de exclusividade.

Sumário

- Criar tabelas
- Criar índices
- Ler documentos
  - Usar parâmetros de consulta
- Inserir documentos
  - Como inserir vários documentos em uma instrução
- Como atualizar documentos
- Como excluir documentos
- Como executar várias instruções em uma transação
- Lógica de novas tentativas
- Implementação de restrições de exclusividade

Criar tabelas

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 }
}
```

Usar a biblioteca ION

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"
    // }
}
```

Criar índices

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 }
}
```

Usar a biblioteca ION

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"
    // }
}
```

Ler documentos

```
// 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**
>
> Quando você executa uma consulta sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as consultas podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Usar parâmetros de consulta

O exemplo de código a seguir usa um parâmetro de consulta do tipo 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.
}
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta do tipo 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.
}
```

O exemplo de código a seguir usa uma matriz de consulta do tipo 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.
}
```

O exemplo de código a seguir usa uma lista C# como valor.

```
// 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
```

```
    //     },
    //     ]
    // }
}
```

Usar a biblioteca ION

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**
>
> Quando você executa uma consulta sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as consultas podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

O exemplo de código a seguir usa um parâmetro de consulta do tipo 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.
```

```
}
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta.

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.
}
```

O exemplo de código a seguir usa uma lista de parâmetros de consulta.

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.
}
```

O exemplo de código a seguir usa uma lista Ion como valor. Para saber mais sobre os diferentes tipos de Ion, consulte Trabalhando com os tipos de dados Amazon Ion na 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"
    //     }
    //     ]
    // }
}
```

Inserir documentos

Os exemplos de código a seguir inserem os tipos de dados Ion.

```
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));
});
```

Usar a biblioteca ION

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);
});
```

Essa transação insere um documento na tabela `Person`. Antes de inserir, ele primeiro verifica se o documento já existe na tabela. Essa verificação torna a transação idempotente por natureza. Mesmo que você execute essa transação várias vezes, ela não causará efeitos colaterais indesejados.

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como inserir vários documentos em uma instrução

Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro C# `List` para a instrução da seguinte maneira.

```
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 }
}
```

Usar a biblioteca ION

Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista Ion para a instrução da seguinte maneira.

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"
    // }
}
```

Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista Ion. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Como atualizar documentos

```
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 }
}
```

Usar a biblioteca ION

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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como excluir documentos

```
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 }
}
```

Usar a biblioteca ION

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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como executar várias instruções em uma transação

```
// 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;
    });
}
```

Usar a biblioteca ION

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;
    });
}
```

Lógica de novas tentativas

Para obter informações sobre a lógica de repetição integrada do driver, consulte Entendendo a política de repetição com o motorista na Amazon QLDB.

Implementação de restrições de exclusividade

QLDBnão oferece suporte a índices exclusivos, mas você pode implementar esse comportamento em seu aplicativo.

Suponha que você queira implementar uma restrição de exclusividade no campo `GovId` da tabela `Person`. Para fazer isso, você pode escrever uma transação que faça o seguinte:

1. Afirme que a tabela não tem documentos existentes com um `GovId` especificado.
2. Insira o documento se a afirmação for aprovada.

Se uma transação concorrente passar simultaneamente pela declaração, somente uma das transações será confirmada com sucesso. A outra transação falhará com uma exceção de OCC conflito.

O exemplo de código a seguir mostra como implementar essa lógica de restrição de exclusividade.

```
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));
});
```

Usar a biblioteca ION

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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como trabalhar com o Amazon Ion

Há várias maneiras de processar dados do Amazon Ion noQLDB. Você pode usar a biblioteca Ion para criar e modificar valores de Ion. Ou você pode usar o mapeador de objetos Ion para mapear CLR objetos antigos simples em C# (POCO) de e para valores de Ion. Versão 1.3.0 do QLDB driver para. NETintroduz suporte para o mapeador de objetos Ion.

As seções a seguir fornecem exemplos de código de processamento de dados Ion usando ambas as técnicas.

Sumário

- Importar o módulo Ion
- Criação de tipos de Ion
- Obter um despejo de binário Ion
- Obter um despejo de texto Ion

Importar o módulo Ion

```
using Amazon.IonObjectMapper;
```

Usar a biblioteca ION

```
using Amazon.IonDotnet.Builders;
```

Criação de tipos de Ion

O exemplo de código a seguir mostra como criar valores de Ion a partir de objetos C# usando o mapeador de objeto Ion.

```
// 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);
```

Usar a biblioteca ION

Os exemplos de código a seguir mostram as duas maneiras de criar valores Ion usando a biblioteca Ion.

Como usar o **`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);
```

Como usar o **`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);
```

Obter um despejo de binário Ion

```
// 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()));
```

Usar a biblioteca ION

```
// 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()));
```

Obter um despejo de texto Ion

```
// 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()));
```

Usar a biblioteca ION

```
// ionObject is an Ion struct
StringWriter sw = new StringWriter();
using (var writer = IonTextWriterBuilder.Build(sw))
{
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(sw.ToString());
```

Para obter mais informações sobre como trabalhar com o Ion, consulte a documentação do Amazon Ion em GitHub. Para obter mais exemplos de código de como trabalhar com o Ion inQLDB, consulteTrabalhando com os tipos de dados Amazon Ion na Amazon QLDB.

QLDBDriver da Amazon para Go

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Para trabalhar com dados em seu livro contábil, você pode se conectar à Amazon a QLDB partir do seu aplicativo Go usando um driver AWS fornecido. Os tópicos a seguir descrevem como começar a usar o QLDB driver do Go.

Tópicos

- Recursos para driver
- Pré-requisitos
- Instalação
- QLDBDriver Amazon para Go — Tutorial de início rápido
- QLDBDriver da Amazon para Go — Referência do livro de receitas

Recursos para driver

Para obter mais informações sobre a funcionalidade suportada pelo driver Go, consulte os recursos a seguir:

- APIreferência: 3.x, 2.x, 1.x
- Código-fonte do driver (GitHub)
- Cookbook Amazon Ion

Pré-requisitos

Antes de começar a usar o QLDB driver do Go, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando a Amazon QLDB. Essa transmissão inclui o seguinte:
    1. Inscreva-se em AWS.
    2. Crie um usuário com as QLDB permissões apropriadas.
    3. Conceda acesso programático para desenvolvimento.
2. (Opcional) Instale um ambiente de desenvolvimento integrado (IDE) de sua escolha. Para obter uma lista dos mais usados IDEs para Go, consulte Plugins do editor e IDEs no site Go.
3. Baixe e instale uma das seguintes versões do Go no site de downloads do Go:
    - 1.15 ou posterior — QLDB driver para Go v3
    - 1.14 — QLDB driver para Go v1 ou v2
4. Configure seu ambiente de desenvolvimento para AWS SDK for Go.
    1. Configure suas AWS credenciais. Recomendamos criar um arquivo de credenciais compartilhadas.

Para obter instruções, consulte Especificação de credenciais no Guia do desenvolvedor do AWS SDK for Go .

2. Defina sua Região da AWS padrão. Para saber como, consulte Especificando Região da AWS.

   Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Em seguida, você pode configurar um aplicativo de exemplo básico e executar exemplos de códigos curtos, ou você pode instalar o driver em um projeto Go existente.

- Para instalar o QLDB driver e o AWS SDK for Go em um projeto existente, vá paraInstalação.
- Para configurar um projeto e executar exemplos de códigos curtos que demonstram transações básicas de dados em um ledger, consulte o Tutorial de início rápido.

Instalação

O QLDB driver para Go é de código aberto no GitHub repositório amazon-qldb-driver-goawslabs/. QLDBsuporta as seguintes versões do driver e suas dependências do Go.

Versão do driver	Versão Go	Status	Data da versão mais recente
1.x	1.14 ou posterior	Lançamento de produção	16 de junho de 2021
2.x	1.14 ou posterior	Lançamento de produção	21 de julho de 2021
3.x	1.15 ou posterior	Lançamento de produção	10 de novembro de 2022

Para instalar o driver

1. Certifique-se de que seu projeto esteja usando módulos Go para instalar as dependências do projeto.

2. No diretório do seu projeto, insira o seguinte comando `go get`.

   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
   ```

A instalação do driver também instala suas dependências, incluindo AWS SDK for Go ou AWS SDK for Go v2 e os pacotes e Amazon Ion.

Para exemplos de código curto de como executar transações básicas de dados em um ledger, consulte o Referência de Cookbook.

QLDBDriver Amazon para Go — Tutorial de início rápido

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você aprende a configurar um aplicativo simples usando a versão mais recente do QLDB driver da Amazon para Go. Este guia inclui etapas para instalar o driver e exemplos de códigos curtos das operações básicas de criação, leitura, atualização e exclusão (CRUD).

Tópicos

- Pré-requisitos
- Etapa 1: Instalar o driver
- Etapa 2: Importar os pacotes
- Etapa 3: Inicializar o driver
- Etapa 4: Crie uma tabela e um índice
- Etapa 5: Inserir um documento

- Etapa 6: consultar o documento
- Etapa 7: Atualize o documento
- Etapa 8: consultar o documento atualizado
- Etapa 9: Descartar a tabela
- Executar o aplicativo completo

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Complete a Pré-requisitos para o driver Go, caso ainda não o tenha feito. Isso inclui se inscrever AWS, conceder acesso programático para desenvolvimento e instalar o Go.
2. Crie um ledger chamado `quick-start`.

   Para saber como criar um ledger, consulte Operações básicas para livros contábeis da Amazon QLDB ou Etapa 1: criar um novo ledger em Conceitos básicos do console.

Etapa 1: Instalar o driver

Certifique-se de que seu projeto esteja usando módulos Go para instalar as dependências do projeto.

No diretório do seu projeto, insira o seguinte comando `go get`.

```
$ go get -u github.com/awslabs/amazon-qldb-driver-go/v3/qldbdriver
```

A instalação do driver também instala suas dependências, incluindo os pacotes AWS SDK for Go v2 e Amazon Ion.

Etapa 2: Importar os pacotes

Importe os seguintes AWS pacotes.

```
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"
)
```

Etapa 3: Inicializar o driver

Inicialize uma instância do driver que se conecta ao ledger chamado `quick-start`.

```
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**
>
> Neste exemplo de código, substitua *us-east-1* com o Região da AWS local onde você criou seu livro contábil.

Etapa 4: Crie uma tabela e um índice

O exemplo de código a seguir mostra como executar as instruções `CREATE TABLE` e `CREATE INDEX`.

```
_, 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)
}
```

Esse código adiciona o código a seguir que cria uma tabela chamada `People` e índices para os campos `firstName` e `age` nessa tabela. Os índices são necessários para otimizar o desempenho da consulta e ajudar a limitar as exceções otimistas de conflitos do controle de simultaneidade (OCC).

Etapa 5: Inserir um documento

Os exemplos de código a seguir mostram como executar uma instrução `INSERT`. QLDBsuporta a linguagem de consulta partiQL (SQLcompatível) e o formato de dados Amazon Ion (superconjunto de). JSON

Uso do PartiQL literal

O código a seguir insere um documento na tabela `People` usando uma instrução partiQL literal de string.

```
_, 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)
}
```

Uso de tipos de dados Ion

Semelhante ao JSONpacote integrado do Go, você pode organizar e desorganizar tipos de dados Go de e para o Ion.

1. Suponha que você tenha a seguinte estrutura Go, chamada `Person`.

```
type Person struct {
    FirstName string `ion:"firstName"`
    LastName  string `ion:"lastName"`
    Age       int    `ion:"age"`
}
```

2. Crie uma instância de `Person`.

```
person := Person{"John", "Doe", 54}
```

O motorista organiza uma representação de `person` de texto codificada por Ion para você.

> **Important**
>
> Para que a organização e desorganização funcionem corretamente, os nomes dos campos da estrutura de dados Go devem ser exportados (primeira letra maiúscula).

3. Passe a instância `person` para o método `Execute` da transação.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People ?", person)
})
if err != nil {
    panic(err)
}
```

Este exemplo usa um ponto de interrogação (?) como um marcador variável para passar as informações do documento para a instrução. Ao usar marcadores, você deve passar um valor de texto codificado por Ion.

> **Tip**
>
> Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista para a instrução da seguinte maneira.
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Etapa 6: consultar o documento

O exemplo de código a seguir mostra como executar uma instrução `SELECT`.

```
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")
}
```

Este exemplo consulta seu documento a partir da tabela `People`, presume que o conjunto de resultados não está vazio e retorna seu documento a partir do resultado.

Etapa 7: Atualize o documento

O exemplo de código a seguir mostra como executar uma instrução `UPDATE`.

```
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)
}
```

Etapa 8: consultar o documento atualizado

O exemplo de código a seguir consulta a tabela `People` por `firstName` e retorna todos os documentos no conjunto de resultados.

```
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")
}
```

Etapa 9: Descartar a tabela

O exemplo de código a seguir mostra como executar uma instrução `DROP TABLE`.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("DROP TABLE People")
})
if err != nil {
```

```
    panic(err)
}
```

Executar o aplicativo completo

O exemplo de código a seguir é a versão completa do aplicativo . Em vez de executar as etapas anteriores individualmente, você também pode copiar e executar esse exemplo de código do início ao fim. Este aplicativo demonstra algumas CRUD operações básicas no livro contábil chamado. `quick-start`

> **Note**
>
> Antes de executar esse código, verifique se você ainda não tem uma tabela ativa chamada `People` no ledger `quick-start`.

```
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)
    }
}
```

QLDBDriver da Amazon para Go — Referência do livro de receitas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este guia de referência mostra casos de uso comuns do QLDB driver Amazon para Go. Ele fornece exemplos de código Go que demonstram como usar o driver para executar operações básicas de criação, leitura, atualização e exclusão (CRUD). Também inclui exemplos de código para processamento de dados do Amazon Ion. Além disso, este guia destaca as práticas recomendadas para tornar as transações idempotentes e implantar restrições de exclusividade.

> **Note**
>
> Quando aplicável, alguns casos de uso têm exemplos de código diferentes para cada versão principal compatível do QLDB driver para Go.

Sumário

- Importação do driver
- Instanciação do driver

- CRUDoperações
  - Criar tabelas
  - Criar índices
  - Ler documentos
    - Usar parâmetros de consulta
  - Inserir documentos
    - Inserindo vários documentos em uma instrução
  - Como atualizar documentos
  - Como excluir documentos
  - Executando várias instruções em uma transação
  - Lógica de novas tentativas
  - Implementação de restrições de exclusividade
- Como trabalhar com o Amazon Ion
  - Importar o módulo Ion
  - Criação de tipos de Ion
  - Obter binário Ion
  - Obter texto Ion

Importação do driver

O exemplo de código a seguir importa o driver e outros AWS pacotes necessários.

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**
>
> Este exemplo também importa o pacote Amazon Ion (`amzn/ion-go/ion`). Você precisa desse pacote para processar dados de íons ao executar algumas operações de dados nesta referência. Para saber mais, consulte Como trabalhar com o Amazon Ion.

Instanciação do driver

O exemplo de código a seguir cria uma instância do driver que se conecta a um nome de ledger especificado em um Região da AWS especificado.

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)
}
```

CRUDoperações

QLDBexecuta operações de criação, leitura, atualização e exclusão (CRUD) como parte de uma transação.

> **Warning**
>
> Como prática recomendada, torne suas transações de gravação estritamente idempotentes.

Tornar as transações idempotentes

Recomendamos que você torne as transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de novas tentativas. Uma transação é idempotente se puder ser executada várias vezes e produzir resultados idênticos a cada vez.

Por exemplo, considere uma transação que insere um documento em uma tabela chamada `Person`. A transação deve primeiro verificar se o documento já existe ou não na tabela. Sem essa verificação, a tabela pode acabar com documentos duplicados.

Suponha que a transação seja confirmada QLDB com sucesso no lado do servidor, mas o cliente expire enquanto espera por uma resposta. Se a transação não for idempotente, o mesmo documento poderá ser inserido mais de uma vez no caso de uma nova tentativa.

Usar índices para evitar varreduras completas da tabela

Também recomendamos executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Sem essa pesquisa indexada, é QLDB necessário fazer uma varredura da tabela, o que pode levar a tempos limite de transação ou conflitos otimistas de controle de simultaneidade (). OCC

Para obter mais informações sobre o OCC, consulte Modelo de QLDB concorrência da Amazon.

Transações criadas implicitamente

A função QLDBDriver.Execute aceita uma função lambda que recebe uma instância de Transaction, que você pode usar para executar instruções. A instância de `Transaction` envolve uma transação criada implicitamente.

Você pode executar instruções na função do Lambda usando a função `Transaction.Execute`. O driver confirma implicitamente a transação quando a função do Lambda retorna.

As seções a seguir mostram como executar CRUD operações básicas, especificar a lógica de repetição personalizada e implementar restrições de exclusividade.

Sumário

- Criar tabelas
- Criar índices
- Ler documentos
  - Usar parâmetros de consulta
- Inserir documentos
  - Inserindo vários documentos em uma instrução
- Como atualizar documentos
- Como excluir documentos
- Executando várias instruções em uma transação
- Lógica de novas tentativas

- Implementação de restrições de exclusividade

Criar tabelas

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE TABLE Person")
})
```

Criar índices

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE INDEX ON Person(GovId)")
})
```

Ler documentos

```
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)
}
```

Usar parâmetros de consulta

O exemplo de código a seguir usa um parâmetro de consulta do tipo nativo.

```
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)
}
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta.

```
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)
}
```

O exemplo de código a seguir usa uma lista de parâmetros de consulta.

```
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**
>
> Quando você executa uma consulta sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Neste exemplo, recomendamos ter um índice no campo `GovId` para

otimizar o desempenho. Sem um índice ativado`GovId`, as consultas podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserir documentos

Os exemplos de código a seguir inserem os tipos de dados nativos.

```
_, 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
})
```

Essa transação insere um documento na tabela `Person`. Antes de inserir, ele primeiro verifica se o documento já existe na tabela. Essa verificação torna a transação idempotente por natureza. Mesmo que você execute essa transação várias vezes, ela não causará efeitos colaterais indesejados.

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserindo vários documentos em uma instrução

Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista para a instrução da seguinte maneira.

```
// people is a list
txn.Execute("INSERT INTO People ?", people)
```

Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Como atualizar documentos

Os exemplos de código a seguir usam tipos de dados nativos.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", "John",
 "TOYENC486FH")
})
```

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como excluir documentos

Os exemplos de código a seguir usam tipos de dados nativos.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("DELETE FROM Person WHERE GovId = ?", "TOYENC486FH")
})
```

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Executando várias instruções em uma transação

```
// 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
}
```

Lógica de novas tentativas

A `Execute` função do driver tem um mecanismo de repetição integrado que repete a transação se ocorrer uma exceção que possa ser repetida (como tempos limite ou conflitos). OCC O número máximo de tentativas de repetição e a estratégia de recuo são configuráveis.

O limite padrão de repetição é`4`, e a estratégia de recuo padrão é ExponentialBackoffStrategycom uma base de `10` milissegundos. Você pode definir a política de repetição por instância de driver e também por transação usando uma instância de RetryPolicy.

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância do driver.

```
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)
  }
}
```

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância anônima particular. A função `SetRetryPolicy` substitui a política de repetição definida para a instância do driver.

```
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")
  })
}
```

Implementação de restrições de exclusividade

QLDBnão oferece suporte a índices exclusivos, mas você pode implementar esse comportamento em seu aplicativo.

Suponha que você queira implementar uma restrição de exclusividade no campo `GovId` da tabela `Person`. Para fazer isso, você pode escrever uma transação que faça o seguinte:

1. Afirme que a tabela não tem documentos existentes com um `GovId` especificado.
2. Insira o documento se a afirmação for aprovada.

Se uma transação concorrente passar simultaneamente pela declaração, somente uma das transações será confirmada com sucesso. A outra transação falhará com uma exceção de OCC conflito.

O exemplo de código a seguir mostra como implementar essa lógica de restrição de exclusividade.

```
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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como trabalhar com o Amazon Ion

As seções a seguir mostram como usar o módulo Amazon Ion para processar dados do Ion.

Sumário

- Importar o módulo Ion
- Criação de tipos de Ion
- Obter binário Ion
- Obter texto Ion

Importar o módulo Ion

```
import "github.com/amzn/ion-go/ion"
```

Criação de tipos de Ion

Atualmente, a biblioteca Ion para Go não oferece suporte ao Document Object Model (DOM), portanto, você não pode criar tipos de dados Ion. Mas você pode organizar e desorganizar entre os tipos nativos de Go e o binário de íons ao trabalhar com. QLDB

Obter binário Ion

```
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]
```

Obter texto Ion

```
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"}
```

Para obter mais informações sobre o Ion, consulte a documentação do Amazon Ion em GitHub. Para obter mais exemplos de código de como trabalhar com o Ion inQLDB, consulteTrabalhando com os tipos de dados Amazon Ion na Amazon QLDB.

QLDBDriver da Amazon para Node.js

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Para trabalhar com dados em seu livro contábil, você pode se conectar à Amazon a QLDB partir do seu aplicativo Node.js usando um driver AWS fornecido. Os tópicos a seguir descrevem como começar a usar o QLDB driver para Node.js.

Tópicos

- Recursos para driver
- Pré-requisitos
- Instalação
- Recomendações de configuração
- QLDBDriver Amazon para Node.js — Tutorial de início rápido
- QLDBDriver da Amazon para Node.js — Referência do livro de receitas

Recursos para driver

Para obter mais informações sobre a funcionalidade suportada pelo driver Node.js, consulte os recursos a seguir:

- APIreferência: 3.x, 2.x, 1.x
- Código-fonte do driver (GitHub)
- Exemplo de código-fonte do aplicativo (GitHub)
- Exemplos de código do Amazon Ion
- Crie uma CRUD operação simples e um fluxo de dados QLDB usando AWS Lambda (AWS Blog)

Pré-requisitos

Antes de começar a usar o QLDB driver para Node.js, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando a Amazon QLDB. Essa transmissão inclui o seguinte:
    1. Inscreva-se em AWS.
    2. Crie um usuário com as QLDB permissões apropriadas.
    3. Conceda acesso programático para desenvolvimento.
2. Instale o Node.js versão 14.x ou posterior a partir do site de downloads do Node.js. (As versões anteriores do driver oferecem suporte ao Node.js versão 10.x ou posterior.)
3. Configure seu ambiente de desenvolvimento AWS SDKpara o for JavaScript em Node.js:
    1. Configure suas AWS credenciais. Recomendamos criar um arquivo de credenciais compartilhadas.

       Para obter mais informações, consulte Carregamento de credenciais do arquivo de credenciais compartilhadas no Guia do desenvolvedor do AWS SDK for JavaScript .
    2. Defina sua Região da AWS padrão. Para aprender como fazer isso, consulte Definindo o Região da AWS.

       Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Em seguida, você pode baixar o aplicativo completo de amostra do tutorial ou instalar somente o driver em um projeto Node.js e executar exemplos de códigos curtos.

- Para instalar o QLDB driver e o AWS SDK for JavaScript no Node.js em um projeto existente, vá paraInstalação.
- Para configurar um projeto e executar exemplos de códigos curtos que demonstram transações básicas de dados em um ledger, consulte o Tutorial de início rápido.
- Para executar exemplos mais detalhados de API operações de dados e gerenciamento no aplicativo de amostra completo do tutorial, consulte o. Tutorial Node.js

Instalação

QLDBsuporta as seguintes versões do driver e suas dependências do Node.js.

Versão do driver	Versão do Node.js	Status	Data da versão mais recente
1.x	10.x ou posterior	Lançamento de produção	5 de junho de 2020
2.x	10.x ou posterior	Lançamento de produção	6 de maio de 2021
3.x	14.x ou posterior	Lançamento de produção	10 de novembro de 2023

Para instalar o QLDB driver usando o npm (o gerenciador de pacotes do Node.js), digite o comando a seguir no diretório raiz do seu projeto.

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
```

O pacote tem dependências pares nos pacotes a seguir. Você também deve instalar esses pacotes como dependências em seu projeto.

3.x

QLDBCliente agregado modular (gerenciamentoAPI)

```
npm install @aws-sdk/client-qldb
```

Cliente modular de QLDBsessão agregada (dadosAPI)

```
npm install @aws-sdk/client-qldb-session
```

Formato de dados Amazon Ion

```
npm install ion-js
```

JavaScript Implementação pura de `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Formato de dados Amazon Ion

```
npm install ion-js@4.0.0
```

JavaScript Implementação pura de `BigInt`

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Formato de dados Amazon Ion

```
npm install ion-js@4.0.0
```

JavaScript Implementação pura de `BigInt`

```
npm install jsbi@3.1.1
```

Como usar o driver para se conectar a um ledger

Depois, você pode importar o driver e usá-lo para se conectar a um ledger. O exemplo de TypeScript código a seguir mostra como criar uma instância de driver para um nome de livro contábil especificado e. Região da AWS

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);
});
```

Para exemplos de código curto de como executar transações básicas de dados em um ledger, consulte o Referência de Cookbook.

Recomendações de configuração

Reutilizar conexões com Keep-alive

QLDBControlador Node.js v3

O HTTPS agente Node.jsHTTP/padrão cria uma nova TCP conexão para cada nova solicitação. Para evitar o custo de estabelecer uma nova conexão, a AWS SDK for JavaScript v3 reutiliza TCP as conexões por padrão. Para obter mais informações e saber como desativar a reutilização de conexões, consulte Reusing connections with keep-alive in Node.js no Guia do desenvolvedor do AWS SDK for JavaScript .

Recomendamos usar a configuração padrão para reutilizar conexões no QLDB driver do Node.js. Durante a inicialização do driver, defina a HTTP opção de cliente de baixo nível `maxSockets` com o mesmo valor que você definiu. `maxConcurrentTransactions`

Por exemplo, consulte o TypeScript código JavaScript a seguir.

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);
```

QLDBControlador Node.js v2

O HTTPS agente Node.jsHTTP/padrão cria uma nova TCP conexão para cada nova solicitação. Para evitar o custo de estabelecer uma nova conexão, recomendamos reutilizar uma conexão existente.

Para reutilizar conexões no QLDB driver para Node.js, use uma das seguintes opções:

- Durante a inicialização do driver, defina as seguintes opções de cliente HTTP de baixo nível:
  - `keepAlive` – `true`
  - `maxSockets`: o mesmo valor que você definiu para `maxConcurrentTransactions`

Por exemplo, consulte o TypeScript código JavaScript a seguir.

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);
```

- Como alternativa, você pode definir as variáveis de ambiente `AWS_NODEJS_CONNECTION_REUSE_ENABLED` para 1. Para obter mais informações, consulte Reutilizar conexões com Keep-Alive em Node.js no Guia do desenvolvedor do AWS SDK for JavaScript .

> **Note**
>
> Se você definir essa variável de ambiente, ela afetará todos os Serviços da AWS que usam o AWS SDK for JavaScript.

QLDBDriver Amazon para Node.js — Tutorial de início rápido

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você aprende a configurar um aplicativo simples usando o QLDB driver da Amazon para Node.js. Este guia inclui etapas para instalar o driver e exemplos curtos JavaScript e de TypeScript código das operações básicas de criação, leitura, atualização e exclusão (CRUD). Para obter exemplos mais detalhados que demonstram essas operações em um aplicativo de amostra completo, consulte o Tutorial Node.js.

> **Note**
>
> Quando aplicável, algumas etapas têm exemplos de código diferentes para cada versão principal compatível do QLDB driver para Node.js.

Tópicos

- Pré-requisitos
- Etapa 1: configurar o projeto do
- Etapa 2: Inicializar o driver
- Etapa 3: Crie uma tabela e um índice
- Etapa 4: Inserir um documento
- Etapa 5: consultar o documento
- Etapa 6: Atualize o documento
- Executar o aplicativo completo

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Complete a Pré-requisitos para o driver Node.js, caso ainda não o tenha feito. Isso inclui se inscrever AWS, conceder acesso programático para desenvolvimento e instalar o Node.js.
2. Crie um ledger chamado `quick-start`.

   Para saber como criar um ledger, consulte Operações básicas para livros contábeis da Amazon QLDB ou Etapa 1: criar um novo ledger em Conceitos básicos do console.

Se você estiver usando TypeScript, você também deve seguir as etapas de configuração a seguir.

Usando TypeScript

Para instalar TypeScript

1. Instale o TypeScript pacote. O QLDB driver é executado em TypeScript 3.8.x.

```
$ npm install --global typescript@3.8.0
```

2. Depois que o pacote for instalado, execute o comando a seguir para garantir que o TypeScript compilador esteja instalado.

```
$ tsc --version
```

Para executar o código nas etapas a seguir, observe que você deve primeiro transpilar seu TypeScript arquivo em JavaScript código executável, da seguinte maneira.

```
$ tsc app.ts; node app.js
```

Etapa 1: configurar o projeto do

Primeiro, configure seu projeto Node.js.

1. Crie uma pasta para o seu aplicativo.

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

2. Para inicializar seu projeto, digite o comando `npm` a seguir e responda às perguntas feitas durante a configuração. Você pode usar padrões para a maioria das perguntas.

```
$ npm init
```

3. Instale o QLDB driver da Amazon para Node.js.
   - Usar a versão 3.x

```
$ npm install amazon-qldb-driver-nodejs --save
```

   - Usar a versão 2.x

```
$ npm install amazon-qldb-driver-nodejs@2.2.0 --save
```

   - Usar a versão 1.x

```
$ npm install amazon-qldb-driver-nodejs@1.0.0 --save
```

4. Instale as dependências de pares do driver.
   - Usar a versão 3.x

```
$ npm install @aws-sdk/client-qldb-session --save
$ npm install ion-js --save
$ npm install jsbi --save
```

- Usar a versão 2.x ou 1.x

```
$ npm install aws-sdk --save
$ npm install ion-js@4.0.0 --save
$ npm install jsbi@3.1.1 --save
```

5. Crie um novo arquivo com `app.js` o nome JavaScript de ou `app.ts` para TypeScript.

   Em seguida, adicione incrementalmente os exemplos de código nas etapas a seguir para experimentar algumas CRUD operações básicas. Ou você pode pular o step-by-step tutorial e, em vez disso, executar o aplicativo completo.

Etapa 2: Inicializar o driver

Inicialize uma instância do driver que se conecta ao ledger chamado `quick-start`. Adicione o seguinte código ao seu arquivo `app.js` ou `app.ts`.

Usar a versão 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**
>
> - Neste exemplo de código, substitua *us-east-1* com o Região da AWS local onde você criou seu livro contábil.
> - Para simplificar, os exemplos de código restantes neste guia usam um driver com configurações padrão, conforme especificado no exemplo a seguir para a versão 1.x. Também é possível usar sua própria instância de driver com uma `RetryConfig` personalizada.

Usar a versão 2.x

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

- Neste exemplo de código, substitua *us-east-1* com o Região da AWS local onde você criou seu livro contábil.

- A versão 2.x introduz o novo parâmetro opcional `RetryConfig` para inicialização `QldbDriver`.
- Para simplificar, os exemplos de código restantes neste guia usam um driver com configurações padrão, conforme especificado no exemplo a seguir para a versão 1.x. Também é possível usar sua própria instância de driver com uma `RetryConfig` personalizada.
- Este exemplo de código inicializa um driver que reutiliza conexões existentes definindo as opções keep-alive. Para saber mais, consulte Recomendações de configuração para o driver Node.js.

Usar a versão 1.x

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**
>
> Você pode definir a variável de ambiente `AWS_REGION` para especificar a Região. Para obter mais informações, consulte Configurar a Região da AWS no Guia do desenvolvedor do AWS SDK for JavaScript .

Etapa 3: Crie uma tabela e um índice

Os exemplos de código a seguir mostram como executar as instruções `CREATE TABLE` e `CREATE INDEX`.

1. Adicione a função a seguir que cria uma tabela chamada `People`.

   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. Adicione a função a seguir que cria um índice para o campo `firstName` na tabela `People`. Os índices são necessários para otimizar o desempenho da consulta e ajudar a limitar as exceções otimistas de conflitos do controle de simultaneidade (OCC).

   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. Na função `main`, você chama primeiro `createTable` e depois chama `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. Execute o código para criar a tabela e o índice.

   JavaScript

   ```
   $ node app.js
   ```

   TypeScript

   ```
   $ tsc app.ts; node app.js
   ```

Etapa 4: Inserir um documento

O exemplo de código a seguir mostra como executar uma instrução `INSERT`. QLDBsuporta a linguagem de consulta partiQL (SQLcompatível) e o formato de dados Amazon Ion (superconjunto de). JSON

1. Adicione o código a seguir que insere um documento na tabela `People`.

   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);
}
```

Este exemplo usa um ponto de interrogação (?) como um marcador variável para passar as informações do documento para a instrução. O método `execute` suporta valores nos tipos Amazon Ion e nos tipos nativos do Python.

> **Tip**
>
> Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista para a instrução da seguinte maneira.
>
> ```
> // people is a list
> txn.execute("INSERT INTO People ?", people);
> ```
>
> Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

2. Na função `main`, remova as chamadas `createTable` e `createIndex` e adicione uma chamada a `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();
}
```

Etapa 5: consultar o documento

O exemplo de código a seguir mostra como executar uma instrução `SELECT`.

1. Adicione o código a seguir que insere um documento da tabela `People`.

   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. Na função `main`, adicione a seguinte chamada `fetchDocuments` após a chamada para `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();
}
```

Etapa 6: Atualize o documento

O exemplo de código a seguir mostra como executar uma instrução `UPDATE`.

1. Adicione o código a seguir que insere um documento na tabela `People`, mudando de `lastName` para `"Stiles"`.

   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. Na função `main`, adicione a seguinte chamada `updateDocuments` após a chamada para `fetchDocuments`. Em seguida, chame `fetchDocuments` novamente para ver os resultados atualizados.

   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. Execute o código para inserir, consultar e atualizar um documento.

   JavaScript

   ```
   $ node app.js
   ```

   TypeScript

   ```
   $ tsc app.ts; node app.js
   ```

Executar o aplicativo completo

Os exemplos de código a seguir são as versões completas de `app.js` e `app.ts`. Em vez de executar as etapas anteriores individualmente, você também pode copiar e executar esse exemplo de código do início ao fim. Este aplicativo demonstra algumas CRUD operações básicas no livro contábil chamado. `quick-start`

> **Note**
>
> Antes de executar esse código, verifique se você ainda não tem uma tabela ativa chamada `People` no ledger `quick-start`.

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();
}
```

Para executar o aplicativo completo, insira o comando a seguir.

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

QLDBDriver da Amazon para Node.js — Referência do livro de receitas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este guia de referência mostra casos de uso comuns do QLDB driver da Amazon para Node.js. Ele fornece JavaScript exemplos de TypeScript código que demonstram como usar o driver para executar operações básicas de criação, leitura, atualização e exclusão (CRUD). Também inclui exemplos de código para processamento de dados do Amazon Ion. Além disso, este guia destaca as práticas recomendadas para tornar as transações idempotentes e implantar restrições de exclusividade.

Sumário

- Importação do driver
- Instanciação do driver
- CRUDoperações
  - Criar tabelas
  - Criar índices
  - Ler documentos
    - Usar parâmetros de consulta
  - Inserir documentos
    - Inserindo vários documentos em uma instrução
  - Como atualizar documentos
  - Como excluir documentos
  - Executando várias instruções em uma transação
  - Lógica de novas tentativas
  - Implementação de restrições de exclusividade
- Como trabalhar com o Amazon Ion
  - Importar o módulo Ion
  - Criação de tipos de Ion
  - Obter um despejo de binário Ion
  - Obter um despejo de texto Ion

Importação do driver

O exemplo de código a seguir importa o driver.

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**
>
> Este exemplo também importa o pacote Amazon Ion (`ion-js`). Você precisa desse pacote para processar dados de íons ao executar algumas operações de dados nesta referência. Para saber mais, consulte Como trabalhar com o Amazon Ion.

Instanciação do driver

O exemplo de código a seguir cria uma instância do driver que se conecta a um nome de ledger especificado usando as configurações padrão.

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

CRUDoperações

QLDBexecuta operações de criação, leitura, atualização e exclusão (CRUD) como parte de uma transação.

> **Warning**
>
> Como prática recomendada, torne suas transações de gravação estritamente idempotentes.

Tornar as transações idempotentes

Recomendamos que você torne as transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de novas tentativas. Uma transação é idempotente se puder ser executada várias vezes e produzir resultados idênticos a cada vez.

Por exemplo, considere uma transação que insere um documento em uma tabela chamada `Person`. A transação deve primeiro verificar se o documento já existe ou não na tabela. Sem essa verificação, a tabela pode acabar com documentos duplicados.

Suponha que a transação seja confirmada QLDB com sucesso no lado do servidor, mas o cliente expire enquanto espera por uma resposta. Se a transação não for idempotente, o mesmo documento poderá ser inserido mais de uma vez no caso de uma nova tentativa.

Usar índices para evitar varreduras completas da tabela

Também recomendamos executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Sem essa pesquisa indexada, é QLDB necessário fazer uma varredura da tabela, o que pode levar a tempos limite de transação ou conflitos otimistas de controle de simultaneidade (). OCC

Para obter mais informações sobre o OCC, consulte Modelo de QLDB concorrência da Amazon.

Transações criadas implicitamente

QldbDriverA. executeLambdaO método aceita uma função lambda que recebe uma instância de TransactionExecutor, que você pode usar para executar instruções. A instância de `TransactionExecutor` envolve uma transação criada implicitamente.

Você pode executar instruções na função do Lambda usando o método de execução da transação. O driver confirma implicitamente a transação quando a função do Lambda retorna.

> **Note**
>
> O método `execute` suporta valores nos tipos Amazon Ion e nos tipos nativos do Node.js. Se você passar um tipo nativo do Node.js como argumento para `execute`, o driver o converterá em um tipo Ion usando o módulo `ion-js` (desde que a conversão para o determinado tipo de dados do Node.js seja suportada). Para ver os tipos de dados e as regras de conversão compatíveis, consulte o Ion JavaScript DOM README.

As seções a seguir mostram como executar CRUD operações básicas, especificar a lógica de repetição personalizada e implementar restrições de exclusividade.

Sumário

- Criar tabelas
- Criar índices
- Ler documentos
  - Usar parâmetros de consulta
- Inserir documentos
  - Inserindo vários documentos em uma instrução
- Como atualizar documentos
- Como excluir documentos
- Executando várias instruções em uma transação
- Lógica de novas tentativas
- Implementação de restrições de exclusividade

Criar tabelas

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');
    });
}());
```

Criar índices

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)');
    });
}());
```

Ler documentos

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']
        }
    });
}());
```

Usar parâmetros de consulta

O exemplo de código a seguir usa um parâmetro de consulta do tipo nativo.

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']
        }
    });
}());
```

O exemplo de código a seguir usa um parâmetro de consulta do tipo 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']
        }
    });
}());
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta.

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']
        }
    });
}());
```

O exemplo de código a seguir usa uma lista de parâmetros de consulta.

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**
>
> Quando você executa uma consulta sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as consultas podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserir documentos

Os exemplos de código a seguir inserem os tipos de dados nativos.

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);
        }
    });
}());
```

Os exemplos de código a seguir inserem os tipos de dados Ion.

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);
        }
    });
}());
```

Essa transação insere um documento na tabela `Person`. Antes de inserir, ele primeiro verifica se o documento já existe na tabela. Essa verificação torna a transação idempotente por natureza. Mesmo que você execute essa transação várias vezes, ela não causará efeitos colaterais indesejados.

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserindo vários documentos em uma instrução

Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista para a instrução da seguinte maneira.

```
// people is a list
txn.execute("INSERT INTO People ?", people);
```

Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Como atualizar documentos

Os exemplos de código a seguir usam tipos de dados nativos.

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');
    });
}());
```

O exemplo de código a seguir usa os tipos de dados Ion.

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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como excluir documentos

Os exemplos de código a seguir usam tipos de dados nativos.

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');
    });
}());
```

O exemplo de código a seguir usa os tipos de dados Ion.

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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Executando várias instruções em uma transação

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;
    });
};
```

Lógica de novas tentativas

O `executeLambda` método do driver tem um mecanismo de repetição integrado que repete a transação se ocorrer uma exceção que pode ser repetida (como tempos limite ou conflitos). OCC O número máximo de tentativas de repetição e a estratégia de recuo são configuráveis.

O limite padrão de repetição é`4`, e a estratégia de recuo padrão é defaultBackoffFunctioncom uma base de `10` milissegundos. Você pode definir a configuração de repetição por instância de driver e também por transação usando uma instância de RetryConfig.

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância do driver.

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);
```

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância do lambda. Essa configuração para `executeLambda` substitui a lógica de repetição definida para a instância do driver.

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);
}());
```

Implementação de restrições de exclusividade

QLDBnão oferece suporte a índices exclusivos, mas você pode implementar esse comportamento em seu aplicativo.

Suponha que você queira implementar uma restrição de exclusividade no campo `GovId` da tabela `Person`. Para fazer isso, você pode escrever uma transação que faça o seguinte:

1. Afirme que a tabela não tem documentos existentes com um `GovId` especificado.
2. Insira o documento se a afirmação for aprovada.

Se uma transação concorrente passar simultaneamente pela declaração, somente uma das transações será confirmada com sucesso. A outra transação falhará com uma exceção de OCC conflito.

O exemplo de código a seguir mostra como implementar essa lógica de restrição de exclusividade.

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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como trabalhar com o Amazon Ion

As seções a seguir mostram como usar o módulo Amazon Ion para processar dados do Ion.

Sumário

- Importar o módulo Ion
- Criação de tipos de Ion
- Obter um despejo de binário Ion
- Obter um despejo de texto Ion

Importar o módulo Ion

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

Criação de tipos de Ion

O exemplo de código a seguir cria um objeto Ion a partir do texto Ion.

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']
```

O exemplo de código a seguir cria um objeto Ion a partir de um dicionário Node.js.

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']
```

Obter um despejo de binário Ion

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
```

Obter um despejo de texto Ion

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"}
```

Para obter mais informações sobre o Ion, consulte a documentação do Amazon Ion em GitHub. Para obter mais exemplos de código de como trabalhar com o Ion inQLDB, consulteTrabalhando com os tipos de dados Amazon Ion na Amazon QLDB.

QLDBDriver Amazon para Python

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Para trabalhar com dados em seu livro contábil, você pode se conectar à Amazon a QLDB partir do seu aplicativo Python usando AWS um driver fornecido. Os tópicos a seguir descrevem como começar a usar o QLDB driver para Python.

Tópicos

- Recursos para driver
- Pré-requisitos
- Instalação
- QLDBDriver Amazon para Python — Tutorial de início rápido
- QLDBDriver Amazon para Python — Referência do livro de receitas

Recursos para driver

Para obter mais informações sobre a funcionalidade suportada pelo driver Python, consulte os recursos a seguir:

- APIreferência: 3.x, 2.x
- Código-fonte do driver (GitHub)
- Exemplo de código-fonte do aplicativo (GitHub)
- Exemplos de código do Amazon Ion

Pré-requisitos

Antes de começar a usar o QLDB driver para Python, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando a Amazon QLDB. Essa transmissão inclui o seguinte:
    1. Inscreva-se em AWS.
    2. Crie um usuário com as QLDB permissões apropriadas.
    3. Conceda acesso programático para desenvolvimento.
2. Instale uma das seguintes versões do Python no site de downloads do Python:
    - 3.6 ou posterior — QLDB driver para Python v3
    - 3.4 ou posterior — QLDB driver para Python v2
3. Configure suas AWS credenciais e seu padrão Região da AWS. Para obter instruções, consulte Início rápido na AWS SDK for Python (Boto3) documentação.

Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Em seguida, você pode baixar o aplicativo completo de amostra do tutorial ou instalar somente o driver em um projeto Python e executar exemplos de códigos curtos.

- Para instalar o QLDB driver e o AWS SDK for Python (Boto3) em um projeto existente, vá paraInstalação.
- Para configurar um projeto e executar exemplos de códigos curtos que demonstram transações básicas de dados em um ledger, consulte o Tutorial de início rápido.
- Para executar exemplos mais detalhados de API operações de dados e gerenciamento no aplicativo de amostra completo do tutorial, consulte o. Tutorial do Python

Instalação

QLDBsuporta as seguintes versões do driver e suas dependências do Python.

Versão do driver	Versão do Python	Status	Data da versão mais recente
2.x	3.4 ou posterior	Lançamento de produção	7 de maio de 2020
3.x	3.6 ou posterior	Lançamento de produção	28 de outubro de 2021

Para instalar o QLDB driver do PyPI `pip` usando (um gerenciador de pacotes para Python), digite o seguinte na linha de comando.

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

A instalação do driver também instala suas dependências, incluindo os pacotes AWS SDK for Python (Boto3) e Amazon Ion.

Como usar o driver para se conectar a um ledger

Depois, você pode importar o driver e usá-lo para se conectar a um ledger. Os exemplos de código Python a seguir mostram como criar uma sessão para um nome de livro contábil especificado.

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)
```

Para exemplos de código curto de como executar transações básicas de dados em um ledger, consulte o Referência de Cookbook.

QLDBDriver Amazon para Python — Tutorial de início rápido

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você aprende a configurar um aplicativo simples usando a versão mais recente do QLDB driver da Amazon para Python. Este guia inclui etapas para instalar o driver e exemplos de códigos curtos das operações básicas de criação, leitura, atualização e exclusão (CRUD). Para obter exemplos mais detalhados que demonstram essas operações em um aplicativo de amostra completo, consulte o Tutorial do Python.

Tópicos

- Pré-requisitos
- Etapa 1: configurar o projeto do
- Etapa 2: Inicializar o driver
- Etapa 3: Crie uma tabela e um índice
- Etapa 4: Inserir um documento
- Etapa 5: consultar o documento
- Etapa 6: Atualize o documento
- Executar o aplicativo completo

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Complete a Pré-requisitos Para o driver Python, caso ainda não o tenha feito. Isso inclui se inscrever AWS, conceder acesso programático para desenvolvimento e instalar o Python versão 3.6 ou posterior.
2. Crie um ledger chamado `quick-start`.

   Para saber como criar um ledger, consulte Operações básicas para livros contábeis da Amazon QLDB ou Etapa 1: criar um novo ledger em Conceitos básicos do console.

Etapa 1: configurar o projeto do

Primeiro, configure seu projeto Python.

> **Note**
>
> Se você usa um IDE que tenha recursos para automatizar essas etapas de configuração, você pode pular para. Etapa 2: Inicializar o driver

1. Crie uma pasta para o seu aplicativo.

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

2. Para instalar o QLDB driver para Python a partir do PyPI, digite o seguinte comando. `pip`

```
$ pip install pyqldb
```

   A instalação do driver também instala suas dependências, incluindo os pacotes AWS SDK for Python (Boto3) e Amazon Ion.

3. Crie um novo arquivo chamado `app.py`.

   Em seguida, adicione incrementalmente os exemplos de código nas etapas a seguir para experimentar algumas CRUD operações básicas. Ou você pode pular o step-by-step tutorial e, em vez disso, executar o aplicativo completo.

Etapa 2: Inicializar o driver

Inicialize uma instância do driver que se conecta ao ledger chamado `quick-start`. Adicione o seguinte código ao arquivo `app.py`.

```
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)
```

Etapa 3: Crie uma tabela e um índice

O exemplo de código a seguir mostra como executar as instruções `CREATE TABLE` e `CREATE INDEX`.

Adicione o código a seguir que cria uma tabela chamada `People` e um índice para o campo `lastName` nessa tabela. Os índices são necessários para otimizar o desempenho da consulta e ajudar a limitar as exceções otimistas de conflitos do controle de simultaneidade (OCC).

```
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))
```

Etapa 4: Inserir um documento

O exemplo de código a seguir mostra como executar uma instrução `INSERT`. QLDBsuporta a linguagem de consulta partiQL (SQLcompatível) e o formato de dados Amazon Ion (superconjunto de). JSON

Adicione o código a seguir que insere um documento na tabela `People`.

```
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))
```

Este exemplo usa um ponto de interrogação (?) como um marcador variável para passar as informações do documento para a instrução. O método `execute_statement` suporta valores nos tipos Amazon Ion e nos tipos nativos do Python.

> **Tip**
>
> Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista para a instrução da seguinte maneira.
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Etapa 5: consultar o documento

O exemplo de código a seguir mostra como executar uma instrução `SELECT`.

Adicione o código a seguir que insere um documento da tabela `People`.

```
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))
```

Etapa 6: Atualize o documento

O exemplo de código a seguir mostra como executar uma instrução `UPDATE`.

1. Adicione o código a seguir que insere um documento na tabela `People`, atualizando `age` para `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. Consulte a tabela novamente para ver o valor atualizado.

```
# Query the updated document
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

3. Para executar o aplicativo, insira o comando a seguir do diretório do seu projeto.

```
$ python app.py
```

Executar o aplicativo completo

O exemplo de código a seguir é a versão completa do aplicativo `app.py`. Em vez de executar as etapas anteriores individualmente, você também pode copiar e executar esse exemplo de código do início ao fim. Este aplicativo demonstra algumas CRUD operações básicas no livro contábil chamado. `quick-start`

> **Note**
>
> Antes de executar esse código, verifique se você ainda não tem uma tabela ativa chamada `People` no ledger `quick-start`.

```
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))
```

Para executar o aplicativo completo, insira o comando a seguir do diretório do seu projeto.

```
$ python app.py
```

QLDBDriver Amazon para Python — Referência do livro de receitas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este guia de referência mostra casos de uso comuns do QLDB driver Amazon para Python. Ele fornece exemplos de código Python que demonstram como usar o driver para executar operações básicas de criação, leitura, atualização e exclusão (CRUD). Também inclui exemplos de código para processamento de dados do Amazon Ion. Além disso, este guia destaca as práticas recomendadas para tornar as transações idempotentes e implantar restrições de exclusividade.

> **Note**
>
> Quando aplicável, alguns casos de uso têm exemplos de código diferentes para cada versão principal compatível do QLDB driver para Python.

Sumário

- Importação do driver
- Instanciação do driver
- CRUDoperações
  - Criar tabelas
  - Criar índices
  - Ler documentos
    - Usar parâmetros de consulta
  - Inserir documentos
    - Inserindo vários documentos em uma instrução
  - Como atualizar documentos
  - Como excluir documentos
  - Executando várias instruções em uma transação
  - Lógica de novas tentativas
  - Implementação de restrições de exclusividade
- Como trabalhar com o Amazon Ion
  - Importar o módulo Ion
  - Criação de tipos de Ion
  - Obter um despejo de binário Ion
  - Obter um despejo de texto Ion

Importação do driver

O exemplo de código a seguir importa o driver.

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**
>
> Este exemplo também importa o pacote Amazon Ion (`amazon.ion.simpleion`). Você precisa desse pacote para processar dados de íons ao executar algumas operações de dados nesta referência. Para saber mais, consulte Como trabalhar com o Amazon Ion.

Instanciação do driver

O exemplo de código a seguir cria uma instância do driver que se conecta a um nome de ledger especificado usando as configurações padrão.

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

CRUDoperações

QLDBexecuta operações de criação, leitura, atualização e exclusão (CRUD) como parte de uma transação.

> **Warning**
>
> Como prática recomendada, torne suas transações de gravação estritamente idempotentes.

Tornar as transações idempotentes

Recomendamos que você torne as transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de novas tentativas. Uma transação é idempotente se puder ser executada várias vezes e produzir resultados idênticos a cada vez.

Por exemplo, considere uma transação que insere um documento em uma tabela chamada `Person`. A transação deve primeiro verificar se o documento já existe ou não na tabela. Sem essa verificação, a tabela pode acabar com documentos duplicados.

Suponha que a transação seja confirmada QLDB com sucesso no lado do servidor, mas o cliente expire enquanto espera por uma resposta. Se a transação não for idempotente, o mesmo documento poderá ser inserido mais de uma vez no caso de uma nova tentativa.

Usar índices para evitar varreduras completas da tabela

Também recomendamos executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Sem essa pesquisa indexada, é QLDB necessário fazer uma varredura da tabela, o que pode levar a tempos limite de transação ou conflitos otimistas de controle de simultaneidade (). OCC

Para obter mais informações sobre o OCC, consulte Modelo de QLDB concorrência da Amazon.

Transações criadas implicitamente

O método pyqldb.driver.qldb_driver.execute_lambda aceita uma função do lambda que recebe uma instância de pyqldb.execution.executor.Executor, que você pode usar para executar instruções. A instância de `Executor` envolve uma transação criada implicitamente.

Você pode executar instruções na função do Lambda usando o método do executor da transação execute_statement. O driver confirma implicitamente a transação quando a função do Lambda retorna.

> **Note**
>
> O método `execute_statement` suporta os tipos Amazon Ion e os tipos nativos do Python. Se você passar um tipo nativo do Python como argumento para `execute_statement`, o driver o converterá em um tipo Ion usando o módulo `amazon.ion.simpleion` (desde que a conversão para o determinado tipo de dados do Python seja suportada). Para ver os tipos de dados e as regras de conversão compatíveis, consulte o código-fonte simpleion.

As seções a seguir mostram como executar CRUD operações básicas, especificar a lógica de repetição personalizada e implementar restrições de exclusividade.

Sumário

- Criar tabelas
- Criar índices
- Ler documentos
  - Usar parâmetros de consulta
- Inserir documentos
  - Inserindo vários documentos em uma instrução
- Como atualizar documentos
- Como excluir documentos
- Executando várias instruções em uma transação
- Lógica de novas tentativas
- Implementação de restrições de exclusividade

Criar tabelas

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")

qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

Criar índices

```
def create_index(transaction_executor):
```

```
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Ler documentos

```
# 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))
```

Usar parâmetros de consulta

O exemplo de código a seguir usa um parâmetro de consulta do tipo nativo.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

O exemplo de código a seguir usa um parâmetro de consulta do tipo Ion.

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

O exemplo de código a seguir usa múltiplos parâmetros de consulta.

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

O exemplo de código a seguir usa uma lista de parâmetros de consulta.

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
```

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> Quando você executa uma consulta sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as consultas podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserir documentos

Os exemplos de código a seguir inserem os tipos de dados nativos.

```
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))
```

Os exemplos de código a seguir inserem os tipos de dados Ion.

```
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))
```

Essa transação insere um documento na tabela `Person`. Antes de inserir, ele primeiro verifica se o documento já existe na tabela. Essa verificação torna a transação idempotente por natureza. Mesmo que você execute essa transação várias vezes, ela não causará efeitos colaterais indesejados.

> **Note**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Inserindo vários documentos em uma instrução

Para inserir vários documentos usando uma única instrução INSERT, você pode passar um parâmetro do tipo lista para a instrução da seguinte maneira.

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

Você não coloca o marcador variável (?) entre colchetes angulares duplos (`<<...>>`) ao passar uma lista. Nas instruções manuais do PartiQL, colchetes angulares duplos denotam uma coleção não ordenada conhecida como bolsa.

Como atualizar documentos

Os exemplos de código a seguir usam tipos de dados nativos.

```
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))
```

O exemplo de código a seguir usa os tipos de dados Ion.

```
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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como excluir documentos

Os exemplos de código a seguir usam tipos de dados nativos.

```
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))
```

O exemplo de código a seguir usa os tipos de dados Ion.

```
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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Executando várias instruções em uma transação

```
# 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))
```

Lógica de novas tentativas

O `execute_lambda` método do driver tem um mecanismo de repetição integrado que repete a transação se ocorrer uma exceção que pode ser repetida (como tempos limite ou conflitos). OCC

3.x

O número máximo de tentativas de repetição e a estratégia de recuo são configuráveis.

O limite de repetição padrão é `4`, e a estratégia de recuo padrão é recuo exponencial e Jitter com uma base de `10` milissegundos. Você pode definir a configuração de repetição por instância de driver e também por transação usando uma instância de pyqldb.config.retry_config. RetryConfig.

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância do driver.

```
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)
```

O exemplo de código a seguir especifica a lógica de repetição com um limite de repetição personalizado e uma estratégia de recuo personalizada para uma instância do lambda. Essa configuração para `execute_lambda` substitui a lógica de repetição definida para a instância do driver.

```
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

O número máximo de tentativas de repetição pode ser configurado. Você pode configurar o limite de novas tentativas definindo a propriedade `retry_limit` ao inicializar `PooledQldbDriver`.

O limite padrão de novas tentativas é `4`.

Implementação de restrições de exclusividade

QLDBnão oferece suporte a índices exclusivos, mas você pode implementar esse comportamento em seu aplicativo.

Suponha que você queira implementar uma restrição de exclusividade no campo `GovId` da tabela `Person`. Para fazer isso, você pode escrever uma transação que faça o seguinte:

1. Afirme que a tabela não tem documentos existentes com um `GovId` especificado.
2. Insira o documento se a afirmação for aprovada.

Se uma transação concorrente passar simultaneamente pela declaração, somente uma das transações será confirmada com sucesso. A outra transação falhará com uma exceção de OCC conflito.

O exemplo de código a seguir mostra como implementar essa lógica de restrição de exclusividade.

```
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**
>
> Neste exemplo, recomendamos ter um índice no campo `GovId` para otimizar o desempenho. Sem um índice ativado`GovId`, as declarações podem ter mais latência e também podem levar a exceções de OCC conflito ou tempos limite de transação.

Como trabalhar com o Amazon Ion

As seções a seguir mostram como usar o módulo Amazon Ion para processar dados do Ion.

Sumário

- Importar o módulo Ion
- Criação de tipos de Ion
- Obter um despejo de binário Ion
- Obter um despejo de texto Ion

Importar o módulo Ion

```
import amazon.ion.simpleion as simpleion
```

Criação de tipos de Ion

O exemplo de código a seguir cria um objeto Ion a partir do texto Ion.

```
ion_text = '{GovId: "TOYENC486FH", FirstName: "Brent"}'
ion_obj = simpleion.loads(ion_text)
```

```
print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['Name']) # prints Brent
```

O exemplo de código a seguir cria um objeto Ion a partir de um `dict` Python.

```
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
```

Obter um despejo de binário Ion

```
# 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'
```

Obter um despejo de texto Ion

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

Para obter mais informações sobre como trabalhar com o Ion, consulte a documentação do Amazon Ion em GitHub. Para obter mais exemplos de código de como trabalhar com o Ion inQLDB, consulteTrabalhando com os tipos de dados Amazon Ion na Amazon QLDB.

Entendendo o gerenciamento de sessões com o motorista na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Se você tiver experiência no uso de um sistema de gerenciamento de banco de dados relacional (RDBMS), talvez esteja familiarizado com conexões simultâneas. QLDBnão tem o mesmo conceito de RDBMS conexão tradicional porque as transações são executadas com mensagens de HTTP solicitação e resposta.

EmQLDB, o conceito análogo é uma sessão ativa. Uma sessão é conceitualmente semelhante a um login de usuário — ela gerencia as informações sobre suas solicitações de transação de dados em um ledger. Uma sessão ativa é aquela que está executando ativamente uma transação. Também pode ser uma sessão que concluiu recentemente uma transação em que o serviço prevê que iniciará outra transação imediatamente. QLDBsuporta uma transação em execução ativa por sessão.

O limite de sessões ativas simultâneas por ledger é definido em Cotas e limites na Amazon QLDB. Depois que esse limite for atingido, qualquer sessão que tente iniciar uma transação resultará em um erro (`LimitExceededException`).

Para obter as melhores práticas para configurar um pool de sessões em seu aplicativo usando o QLDB driver, consulte as recomendações Configurando o objeto QldbDriver de QLDB drivers da Amazon.

Sumário

- Ciclo de vida da sessão
- Expiração da sessão
- Tratamento da sessão no QLDB driver
  - Visão geral do agrupamento de sessões
  - Pool de sessões e lógica de transação
  - Retornar as sessões ao pool

Ciclo de vida da sessão

A sequência de API operações de QLDBsessão a seguir representa o ciclo de vida típico de uma QLDB sessão:

1. `StartSession`
2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`

5. Repita as etapas 2 a 4 para iniciar mais transações (uma transação por vez).
6. `EndSession`

Expiração da sessão

QLDBexpira e descarta uma sessão após uma vida útil total de 13 a 17 minutos, independentemente de estar executando ativamente uma transação. As sessões podem ser perdidas ou prejudicadas por vários motivos, como falha de hardware, falha de rede ou reinicialização de aplicativos. QLDBimpõe uma vida útil máxima nas sessões para garantir que o aplicativo cliente seja resiliente a falhas de sessão.

Tratamento da sessão no QLDB driver

Embora você possa usar um AWS SDK para interagir diretamente com a QLDBsessãoAPI, isso aumenta a complexidade e exige que você calcule um resumo de confirmação. QLDBusa esse resumo de confirmação para garantir a integridade da transação. Em vez de interagir diretamente com issoAPI, recomendamos usar o QLDB driver.

O driver fornece uma camada de abstração de alto nível acima dos dados transacionais. API Ele simplifica o processo de execução de instruções partiQL em dados contábeis por meio do gerenciamento de chamadas. SendCommandAPI Essas API chamadas exigem vários parâmetros que o driver manipula para você, incluindo o gerenciamento de sessões, transações e política de repetição em caso de erros.

Tópicos

- Visão geral do agrupamento de sessões
- Pool de sessões e lógica de transação
- Retornar as sessões ao pool

Visão geral do agrupamento de sessões

Nas versões mais antigas do QLDB driver (por exemplo, Java v1.1.0), fornecemos duas implementações do objeto driver: uma padrão, não `QldbDriver` agrupada e uma. `PooledQldbDriver` Como o nome sugere, o `PooledQldbDriver` mantém um pool de sessões que são reutilizadas em todas as transações.

Com base no feedback do usuário, os desenvolvedores preferem usar o atributo de agrupamento e seus benefícios por padrão, em vez de usar o driver padrão. Então, removemos `PooledQldbDriver` e movemos a funcionalidade de agrupamento de sessões para `QldbDriver`. Essa alteração está incluída na versão mais recente de cada driver (por exemplo, Java v2.0.0).

O driver fornece três níveis de abstrações:

- Driver (implementação de driver agrupado): a abstração de nível superior. O driver mantém e gerencia um conjunto de sessões. Quando você solicita que o driver execute uma transação, ele escolhe uma sessão do pool e a usa para executar a transação. Se a transação falhar devido a um erro de sessão (`InvalidSessionException`), o driver usa outra sessão para repetir a transação. Essencialmente, o driver oferece uma experiência de sessão totalmente gerenciada.
- Sessão: um nível abaixo da abstração do driver. Uma sessão está contida em um pool e o driver gerencia o ciclo de vida da sessão. Se uma transação falhar, o driver a repetirá até um número especificado de tentativas usando a mesma sessão. Mas se a sessão encontrar um erro (`InvalidSessionException`), QLDB descarte-o internamente. O driver é então responsável por obter outra sessão do pool para repetir a transação.
- Transação: o nível mais baixo de abstração. Uma sessão está contida em uma sessão e a sessão gerencia o ciclo de vida da transação. A sessão é responsável por repetir a transação em caso de erro. A sessão também garante que não vaze uma transação aberta que não tenha sido confirmada ou cancelada.

Na versão mais recente de cada driver, você pode realizar operações somente no nível de abstração do driver. Você não tem controle direto sobre sessões e transações individuais (ou seja, não há API operações para iniciar manualmente uma nova sessão ou transação).

Pool de sessões e lógica de transação

A versão mais recente de cada driver não fornece mais uma implementação não agrupada do objeto driver. Por padrão, o objeto `QldbDriver` gerencia o pool de sessões. Quando você faz uma chamada para executar uma transação, o driver executa as seguintes etapas:

1. O driver verifica se atingiu o limite do pool de sessões. Nesse caso, o driver imediatamente lança uma exceção `NoSessionAvailable`. Caso contrário, segue para a próxima etapa.
2. O driver verifica se o pool tem uma sessão disponível.
    - Se uma sessão estiver disponível no pool, o driver a usará para executar uma transação.

- Se nenhuma sessão estiver disponível no pool, o driver criará uma nova sessão para executar uma transação.

3. Depois que o driver obtém uma sessão na etapa 2, ele chama a operação `execute` na instância da sessão.
4. Na operação `execute` da sessão, o driver tenta iniciar uma transação chamando `startTransaction`.
   - Se a sessão não for válida, a `startTransaction` chamada falhará e o driver retornará à etapa 1.
   - Se a `startTransaction` chamada for bem-sucedida, o driver prossegue para a próxima etapa.
5. O driver executa sua expressão lambda. Essa expressão lambda pode conter uma ou mais chamadas para executar instruções partiQL. Depois que a expressão lambda termina de ser executada sem erros, o driver continua confirmando a transação.
6. A confirmação da transação pode ter um dos dois resultados:
   - A confirmação é bem-sucedida e o driver retorna o controle ao código do seu aplicativo.
   - A confirmação falha devido a um conflito otimista de controle de concorrência (OCC). Nesse caso, o driver repete as etapas 4 a 6 usando a mesma sessão. O número máximo de tentativas de repetição é configurável no código do aplicativo. O limite padrão é 4.

> **Note**
>
> Se um `InvalidSessionException` for lançado durante as etapas de 4 a 6, o driver marcará a sessão como encerrada e retornará à etapa 1 para repetir a transação.
> Se alguma outra exceção for lançada durante as etapas de 4 a 6, o driver verificará se a exceção pode ser repetida. Nesse caso, ele repete a transação até o número especificado de tentativas de repetição. Caso contrário, ele propaga (emite bolhas e lança) a exceção no código do seu aplicativo.

Retornar as sessões ao pool

Se um `InvalidSessionException` ocorrer a qualquer momento durante o curso de uma transação ativa, o driver não retornará a sessão ao pool. QLDBEm vez disso, descarta a sessão e o driver recebe outra sessão do pool. Em todos os outros casos, o driver retorna a sessão para o pool.

Recomendações QLDB para motoristas da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve as melhores práticas para configurar e usar o QLDB driver da Amazon para qualquer idioma compatível. Os exemplos de código fornecidos são especificamente para Java.

Essas recomendações se aplicam à maioria dos casos de uso comuns, mas um tamanho não serve para todos. Use as recomendações a seguir conforme achar adequado para seu aplicativo.

Tópicos

- Configurando o objeto QldbDriver
- Tentar novamente com exceções
- Otimizar o desempenho
- Como executar várias instruções por transação

Configurando o objeto QldbDriver

O objeto `QldbDriver` gerencia as conexões com seu ledger mantendo um pool de sessões que são reutilizadas em todas as transações. Uma sessão representa uma única conexão com o ledger. QLDBsuporta uma transação em execução ativa por sessão.

> **Important**
>
> Para versões mais antigas do driver, a funcionalidade de agrupamento de sessões ainda está no `PooledQldbDriver` objeto em vez de `QldbDriver`. Se você estiver usando uma das versões a seguir, substitua qualquer menção de `QldbDriver` por `PooledQldbDriver` pelo restante deste tópico.

Driver	Version (Versão)
Java	`1.1.0` ou mais cedo
.NET	`0.1.0-beta`
Node.js	`1.0.0-rc.1` ou mais cedo
Python	`2.0.2` ou mais cedo

O objeto `PooledQldbDriver` está obsoleto na versão mais recente dos drivers. Recomendamos atualizar para a versão mais recente e converter todas as instâncias de `PooledQldbDriver` para`QldbDriver`.

Configurar QldbDriver como um objeto global

Para otimizar o uso de drivers e sessões, certifique-se de que exista apenas uma instância global do driver na instância do seu aplicativo. Por exemplo, em Java, você pode usar frameworks de injeção de dependência, como Spring, Google Guice ou Dagger. O exemplo de código a seguir mostra como configurar `QldbDriver` como singleton.

```
@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();
}
```

Configurar as tentativas de nova tentativa

O driver repete transações automaticamente quando ocorrem exceções transitórias comuns (como `SocketTimeoutException` ou `NoHttpResponseException`). Para definir o número máximo de tentativas de repetição, você pode usar o parâmetro `maxRetries` do objeto da configuração `transactionRetryPolicy` ao criar uma instância do `QldbDriver`. (Para versões mais antigas do driver, conforme listado na seção anterior, use o parâmetro `retryLimit` de `PooledQldbDriver`.)

O valor padrão de `maxRetries` é `4`.

Erros do lado do cliente, como não é possível tentar `InvalidParameterException` novamente. Quando eles ocorrem, a transação é abortada, a sessão é retornada ao pool e a exceção é lançada para o cliente do driver.

Configurar o número máximo de sessões e transações simultâneos

O número máximo de sessões de ledger usadas por uma instância de `QldbDriver` para executar transações é definido por seu parâmetro `maxConcurrentTransactions`. (Para versões mais antigas do driver, conforme listado na seção anterior, isso é definido pelo parâmetro `poolLimit` de `PooledQldbDriver`.)

Esse limite deve ser maior que zero e menor ou igual ao número máximo de HTTP conexões abertas que o cliente da sessão permite, conforme definido pelo específico AWS SDK. Por exemplo, em Java, o número máximo de conexões é definido no ClientConfigurationobjeto.

O valor padrão de `maxConcurrentTransactions` é a configuração máxima de conexão do seu AWS SDK.

Ao configurar o `QldbDriver` em seu aplicativo, considere as seguintes considerações de escalabilidade:

- Seu pool sempre deve ter pelo menos tantas sessões quanto o número de transações em execução simultânea que você planeja ter.
- Em um modelo multiencadeado em que um thread do supervisor delega aos threads do trabalhador, o driver deve ter pelo menos tantas sessões quanto o número de threads do trabalhador. Caso contrário, no pico de carga, os threads estarão esperando na fila por uma sessão disponível.
- O limite de sessões ativas simultâneas por ledger é definido em Cotas e limites na Amazon QLDB. Certifique-se de não ter configurado mais do que esse limite de sessões simultâneas a serem usadas em um único ledger em todos os clientes.

Tentar novamente com exceções

Ao tentar novamente as exceções que ocorrem emQLDB, considere as recomendações a seguir.

Tentando novamente OccConflictException

As exceções de conflito do controle otimista de simultaneidade (OCC) ocorrem quando os dados que a transação está acessando são alterados desde o início da transação. QLDBlança essa exceção ao tentar confirmar a transação. O driver repete a transação até quantas vezes `maxRetries` estiver configurada.

Para obter mais informações OCC e melhores práticas para usar índices para limitar OCC conflitos, consulteModelo de QLDB concorrência da Amazon.

Tentando novamente com outras exceções fora do QldbDriver

Para repetir uma transação fora do driver quando exceções personalizadas definidas pelo aplicativo são lançadas durante o runtime, você deve encapsular a transação. Por exemplo, em Java, o código a seguir mostra como usar a biblioteca Reslience4J para repetir uma transação em. QLDB

```
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**
>
> Tentar novamente uma transação fora do QLDB driver tem um efeito multiplicador. Por exemplo, se `QldbDriver` estiver configurada para repetir três vezes e a lógica de repetição personalizada também tentar três vezes, a mesma transação poderá ser repetida até nove vezes.

Tornar as transações idempotentes

Como prática recomendada, torne suas transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de tentativas repetidas. Uma transação é idempotente se puder ser executada várias vezes e produzir resultados idênticos a cada vez.

Para saber mais, consulte Modelo de QLDB concorrência da Amazon.

Otimizar o desempenho

Para otimizar o desempenho ao executar transações usando o driver, considere as seguintes considerações:

- A `execute` operação sempre faz no mínimo três `SendCommand` API chamadas paraQLDB, incluindo os seguintes comandos:
  1. `StartTransaction`
  2. `ExecuteStatement`

     Esse comando é invocado para cada instrução partiQL que você executa no bloco `execute`.
  3. `CommitTransaction`

  Considere o número total de API chamadas feitas quando você calcula a carga de trabalho geral do seu aplicativo.
- Em geral, recomendamos começar com um gravador de thread único e otimizar as transações agrupando várias declarações em lotes em uma única transação. Maximize as cotas de tamanho da transação, tamanho do documento e número de documentos por transação, conforme definido em Cotas e limites na Amazon QLDB.

- Se o agrupamento em lotes não for suficiente para grandes cargas de transações, você pode tentar o multiencadeamento adicionando gravadores adicionais. No entanto, você deve considerar cuidadosamente os requisitos de seu aplicativo para sequenciamento de documentos e transações e a complexidade adicional que isso introduz.

Como executar várias instruções por transação

Conforme descrito na seção anterior, você pode executar várias declarações por transação para otimizar o desempenho do seu aplicativo. No exemplo de código a seguir, você consulta uma tabela e, em seguida, atualiza um documento nessa tabela dentro de uma transação. Você faz isso passando uma expressão lambda para a operação `execute`.

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))
```

A operação `execute` do driver inicia implicitamente uma sessão e uma transação nessa sessão. Cada instrução que você executa na expressão lambda é encapsulada na transação. Depois que todas as instruções são executadas, o driver confirma automaticamente a transação. Se alguma instrução falhar após o limite de repetição automática ser esgotado, a transação será abortada.

Propagar exceções em uma transação

Ao executar várias instruções por transação, geralmente não recomendamos que você capture e engula exceções dentro da transação.

Por exemplo, em Java, o programa a seguir captura qualquer instância de `RuntimeException`, registra o erro e continua. Esse exemplo de código é considerado uma prática ruim porque a transação é bem-sucedida mesmo quando a instrução UPDATE falha. Portanto, o cliente pode presumir que a atualização foi bem-sucedida quando não foi.

> **Warning**
>
> Não use esse exemplo de código. É fornecido para mostrar um exemplo de antipadrão que é considerado uma má prática.

```
// 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.");
}
```

Em vez disso, propague (aumente) a exceção. Se alguma parte da transação falhar, deixe a operação `execute` abortar a transação para que o cliente possa lidar com a exceção adequadamente.

Entendendo a política de repetição com o motorista na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O QLDB motorista da Amazon usa uma política de repetição para lidar com exceções transitórias ao tentar novamente de forma transparente uma transação com falha. Essas exceções, como `CapacityExceededException` e `RateExceededException`, normalmente se corrigem após um

período de tempo. Se a transação que falhou com a exceção for repetida após um atraso adequado, é provável que seja bem-sucedida. Isso ajuda a melhorar a estabilidade do aplicativo que usaQLDB.

Tópicos

- Tipos de erros que podem ser tentados novamente
- Política padrão de tentativas

Tipos de erros que podem ser tentados novamente

O driver repete automaticamente uma transação se e somente se alguma das seguintes exceções ocorrer durante uma operação dentro dessa transação:

- CapacityExceededException— Retornado quando a solicitação excede a capacidade de processamento do livro contábil.
- InvalidSessionException— Retornado quando uma sessão não é mais válida ou se a sessão não existe.
- LimitExceededException— Retornado se um limite de recursos, como o número de sessões ativas, for excedido.
- OccConflictException— Retornado quando uma transação não pode ser gravada no diário devido a uma falha na fase de verificação do controle otimista de simultaneidade ()OCC.
- RateExceededException— Retornado quando a taxa de solicitações excede a taxa de transferência permitida.

Política padrão de tentativas

A política de repetição consiste em uma condição de nova tentativa e uma estratégia de recuo. A condição de nova tentativa define quando uma transação deve ser repetida, enquanto a estratégia de recuo define quanto tempo esperar antes de repetir a transação.

Ao criar uma instância do driver, a política de repetição padrão especifica a repetição até quatro vezes e o uso de uma estratégia de recuo exponencial. A estratégia de recuo exponencial usa um atraso mínimo de 10 milissegundos e um atraso máximo de 5000 milissegundos, com jitter igual. Se a transação não puder ser confirmada com sucesso dentro da política de repetição, recomendamos tentar a transação em outro momento.

O conceito por detrás do recuo exponencial é usar esperas progressivamente mais longas entre as novas tentativas para respostas de erro consecutivas. Para obter mais informações, consulte a postagem do AWS blog Exponential Backoff and Jitter.

Erros comuns do QLDB driver da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve os erros de tempo de execução que podem ser gerados pelo QLDB driver da Amazon ao interagir com a QLDBsessão API.

Veja a seguir uma lista de exceções comuns retornadas pelo driver. Cada exceção inclui a mensagem de erro específica, seguida por uma breve descrição e sugestões de possíveis soluções.

CapacityExceededException

Mensagem: Capacidade excedida

A Amazon QLDB rejeitou a solicitação porque ela excedeu a capacidade de processamento do livro contábil. QLDBimpõe um limite de escala interno por livro contábil para manter a integridade e o desempenho do serviço. Esse limite varia de acordo com o tamanho da workload de cada solicitação individual. Por exemplo, uma solicitação pode ter uma workload maior se realizar transações de dados ineficientes, como varreduras de tabelas resultantes de uma consulta qualificada não indexada.

Recomendamos que você espere antes de repetir a solicitação. Se seu aplicativo encontrar essa exceção de forma consistente, otimize suas instruções e diminua a taxa e o volume das solicitações que você envia para o ledger. Exemplos de otimização de instruções incluem a execução de menos instruções por transação e o ajuste dos índices da tabela. Para saber como otimizar instruções e evitar varreduras de tabelas, consulte Otimizar a performance da consulta.

Também recomendamos usar a versão mais recente do QLDB driver. O driver tem uma política de repetição padrão que usa recuo exponencial e Jitter para repetir automaticamente exceções

como essa. O conceito por detrás do recuo exponencial é usar esperas progressivamente mais longas entre as novas tentativas para respostas de erro consecutivas.

InvalidSessionException

Mensagem: Transação `transactionId` expirou

Uma transação excedeu sua vida útil máxima. Uma transação pode ser executada por até 30 segundos antes de ser confirmada. Após esse limite de tempo limite, qualquer trabalho realizado na transação é rejeitado e a sessão é QLDB descartada. Esse limite protege o cliente de sessões vazadas ao iniciar transações e não confirmá-las ou cancelá-las.

Se essa for uma exceção comum em seu aplicativo, é provável que as transações estejam simplesmente demorando muito para serem executadas. Se o runtime da transação estiver demorando mais de 30 segundos, otimize seus extratos para acelerar as transações. Exemplos de otimização de instruções incluem a execução de menos instruções por transação e o ajuste dos índices da tabela. Para ter mais informações, consulte Otimizar a performance da consulta.

InvalidSessionException

Mensagem: Sessão `sessionId` expirou

QLDBdescartou a sessão porque ela excedeu sua vida útil total máxima. QLDBdescarta as sessões após 13 a 17 minutos, independentemente de uma transação ativa. As sessões podem ser perdidas ou prejudicadas por vários motivos, como falha de hardware, falha de rede ou reinicialização de aplicativos. Portanto, QLDB impõe uma vida útil máxima às sessões para garantir que o software cliente seja resiliente a falhas de sessão.

Se você encontrar essa exceção, recomendamos que você adquira uma nova sessão e repita a transação. Também recomendamos usar a versão mais recente do QLDB driver, que gerencia o pool de sessões e sua integridade em nome do aplicativo.

InvalidSessionException

Mensagem: Essa sessão não existe

O cliente tentou fazer transações QLDB usando uma sessão que não existe. Supondo que o cliente esteja usando uma sessão que existia anteriormente, a sessão pode não existir mais devido a um dos seguintes motivos:

- Se uma sessão estiver envolvida em uma falha interna do servidor (ou seja, um erro com o código de HTTP resposta 500), QLDB pode optar por descartar a sessão completamente,

em vez de permitir que o cliente faça transações com uma sessão de estado incerto. Então, qualquer nova tentativa nessa sessão falhará com esse erro.

- As sessões expiradas acabam sendo esquecidas porQLDB. Então, qualquer tentativa de continuar usando a sessão resultará nesse erro, em vez do `InvalidSessionException` inicial.

Se você encontrar essa exceção, recomendamos que você adquira uma nova sessão e repita a transação. Também recomendamos usar a versão mais recente do QLDB driver, que gerencia o pool de sessões e sua integridade em nome do aplicativo.

RateExceededException

Mensagem: A taxa foi excedida

QLDBlimitou um cliente com base na identidade do chamador. QLDBimpõe a limitação por região e por conta usando um algoritmo de limitação de token bucket. QLDBfaz isso para ajudar no desempenho do serviço e para garantir o uso justo para todos os QLDB clientes. Por exemplo, tentar adquirir um grande número de sessões simultâneas usando a operação `StartSessionRequest` pode levar ao controle de utilização.

Para manter a integridade do aplicativo e reduzir ainda mais o controle de utilização, você pode tentar novamente essa exceção usando recuo exponencial e jitter. O conceito por detrás do recuo exponencial é usar esperas progressivamente mais longas entre as novas tentativas para respostas de erro consecutivas. Recomendamos usar a versão mais recente do QLDB driver. O driver tem uma política de repetição padrão que usa recuo exponencial e Jitter para repetir automaticamente exceções como essa.

A versão mais recente do QLDB driver também pode ajudar se seu aplicativo estiver constantemente sendo limitado QLDB para `StartSessionRequest` fazer chamadas. O driver mantém um pool de sessões que são reutilizadas em todas as transações, o que pode ajudar a reduzir o número de chamadas `StartSessionRequest` que seu aplicativo faz. Para solicitar um aumento nos limites de API limitação, entre em contato com o AWS Support Centro.

LimitExceededException

Mensagem: Excedeu o limite da sessão

Um ledger excedeu sua cota (também conhecida como limite) no número de sessões ativas. Essa cota é definida em Cotas e limites na Amazon QLDB. Eventualmente, a contagem de sessões ativas de um ledger é consistente, e ledgers consistentemente próximos à cota podem ver essa exceção periodicamente.

Para manter a integridade do seu aplicativo, recomendamos tentar novamente essa exceção. Para evitar essa exceção, certifique-se de não ter configurado mais de 1.500 sessões simultâneas para serem usadas em um único ledger em todos os clientes. Por exemplo, você pode usar o maxConcurrentTransactionsmétodo do QLDBdriver Amazon para Java para configurar o número máximo de sessões disponíveis em uma instância do driver.

QldbClientException

Mensagem: Um resultado transmitido só é válido quando a transação principal está aberta

A transação está fechada e não pode ser usada para recuperar os resultados. QLDB Uma transação é fechada quando é confirmada ou cancelada.

Essa exceção ocorre quando o cliente está trabalhando diretamente com o `Transaction` objeto e está tentando recuperar os resultados QLDB após confirmar ou cancelar uma transação. Para mitigar esse problema, o cliente deve ler os dados antes de fechar a transação.

Começando a usar a Amazon QLDB usando um exemplo de tutorial de aplicativo

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você usa o QLDB driver da Amazon com um AWS SDK para criar um QLDB livro contábil e preenchê-lo com dados de amostra. O driver permite que seu aplicativo interaja com o QLDB uso dos dados transacionais. API O AWS SDK suporta a interação com o gerenciamento de QLDB recursosAPI.

O exemplo de livro contábil que você cria nesse cenário é um banco de dados do departamento de veículos motorizados (DMV) que rastreia as informações históricas completas sobre registros de veículos. Os tópicos a seguir explicam como adicionar registros de veículos, modificá-los e visualizar o histórico de alterações nesses registros. Este guia também mostra como verificar criptograficamente um documento de registro e conclui limpando os recursos e excluindo o ledger de amostra.

Este tutorial de aplicativo de exemplo está disponível para as seguintes linguagens de programação.

Tópicos

- Tutorial QLDB do Amazon Java
- Tutorial QLDB do Amazon Node.js
- Tutorial do Amazon QLDB Python

Tutorial QLDB do Amazon Java

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta implementação do aplicativo de amostra do tutorial, você usa o QLDB driver da Amazon com o AWS SDK for Java para criar um QLDB livro contábil e preenchê-lo com dados de amostra.

Ao trabalhar neste tutorial, você pode consultar a AWS SDK for Java APIReferência para API operações de gerenciamento. Para operações de dados transacionais, você pode consultar o QLDBDriver for Java API Reference.

> **Note**
>
> Quando aplicável, algumas etapas do tutorial têm comandos ou exemplos de código diferentes para cada versão principal compatível do QLDB driver para Java.

Tópicos

- Instalando o aplicativo de amostra Amazon QLDB Java
- Etapa 1: criar um novo ledger
- Etapa 2: Testar a conectividade com o ledger
- Etapa 3: criar tabelas, índices e dados de amostra
- Etapa 4: consultar as tabelas em um ledger
- Etapa 5: Modificar documentos em um ledger

- Etapa 6: Visualizar o histórico de revisão de um documento
- Etapa 7: verificar um documento em um ledger
- Etapa 8: exportar e validar dados do diário em um ledger
- Etapa 9 (opcional): Limpar os recursos

Instalando o aplicativo de amostra Amazon QLDB Java

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve como instalar e executar o aplicativo de QLDB amostra fornecido pela Amazon para o tutorial step-by-step Java. O caso de uso desse exemplo de aplicativo é um banco de dados do departamento de veículos motorizados (DMV) que rastreia as informações históricas completas sobre registros de veículos.

O aplicativo DMV de amostra para Java é de código aberto no GitHub repositório amazon-qldb-dmv-sampleaws-samples/ -java.

Pré-requisitos

Antes de começar, certifique-se de concluir o QLDB driver para JavaPré-requisitos. Essa transmissão inclui o seguinte:

1. Inscreva-se em AWS.
2. Crie um usuário com as QLDB permissões apropriadas. Para concluir todas as etapas deste tutorial, você precisa de acesso administrativo total ao seu recurso contábil por meio do QLDBAPI.
3. Se você estiver usando um que não IDE seja AWS Cloud9, instale o Java e conceda acesso programático para desenvolvimento.

Instalação

As etapas a seguir descrevem como baixar e configurar o aplicativo de amostra com um ambiente de desenvolvimento local. Ou você pode automatizar a configuração do aplicativo de amostra usando

AWS Cloud9 como seu IDE e um AWS CloudFormation modelo para provisionar seus recursos de desenvolvimento.

Seu ambiente de desenvolvimento local

Essas instruções descrevem como baixar e instalar o aplicativo de amostra QLDB Java usando seus próprios recursos e ambiente de desenvolvimento.

Para baixar a aplicação de exemplo

1. Digite o comando a seguir para clonar o aplicativo de GitHub amostra.

   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
   ```

   Esse pacote inclui a configuração do Gradle e o código completo do Tutorial de Java.

2. Carregar e executar o aplicativo oferecido.
   - Se você estiver usando o Eclipse:
     a. Inicie o Eclipse e, no menu Eclipse, escolha Arquivo, Importar e, em seguida, Projeto Gradle existente.
     b. No diretório raiz do projeto, procure e selecione o diretório do aplicativo que contém o arquivo `build.gradle`. Em seguida, escolha Concluir para usar as configurações padrão do Gradle para a importação.
     c. Você pode tentar executar o programa `ListLedgers` como exemplo. Abra o menu de contexto (clique com o botão direito do mouse) do arquivo `ListLedgers.java` e escolha Executar como aplicativo Java.
   - Se você estiver usando IntelliJ:
     a. Inicie o IntelliJ e, no menu IntelliJ, escolha Arquivo e, em seguida, Abrir.

b. No diretório raiz do projeto, procure e selecione o diretório do aplicativo que contém o arquivo `build.gradle`. Escolha OK. Mantenha as configurações padrão e escolha OK novamente.

c. Você pode tentar executar o programa `ListLedgers` como exemplo. Abra o menu de contexto (clique com o botão direito do mouse) do `ListLedgers.java` arquivo e escolha Executar 'ListLedgers'.

3. Continue para Etapa 1: criar um novo ledger para iniciar o tutorial e criar um ledger.

AWS Cloud9

Essas instruções descrevem como automatizar a configuração do aplicativo de amostra de registro de QLDB veículos da Amazon para Java, usando AWS Cloud9como seuIDE. Neste guia, você usa um modelo AWS CloudFormation para provisionar seus recursos de desenvolvimento.

Para obter mais informações sobre AWS Cloud9, consulte o Guia AWS Cloud9 do usuário. Para saber mais sobre AWS CloudFormation, consulte o Guia do usuário de AWS CloudFormation.

Tópicos

- Parte 1: Provisionar seus recursos
- Parte 2: Configure seu IDE
- Parte 3: Executar o aplicativo QLDB DMV de amostra

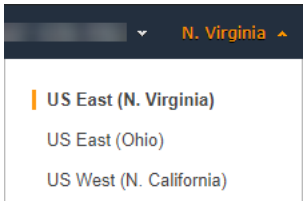
Parte 1: Provisionar seus recursos

Nesta primeira etapa, você usa AWS CloudFormation para provisionar os recursos necessários para configurar seu ambiente de desenvolvimento com o aplicativo de QLDB amostra da Amazon.

Para abrir o AWS CloudFormation console e carregar o modelo QLDB de aplicativo de amostra

1. Faça login no AWS Management Console e abra o AWS CloudFormation console em https://console.aws.amazon.com/cloudformation.

   Mude para uma região que ofereça suporteQLDB. Para obter uma lista completa, consulte os QLDBendpoints e cotas da Amazon no. Referência geral da AWS A captura de tela a seguir AWS Management Console mostra o Leste dos EUA (Norte da Virgínia) como o selecionado Região da AWS.



2. No AWS CloudFormation console, escolha Criar pilha e, em seguida, escolha Com novos recursos (padrão).
3. Na página Criar pilha, em Especificar modelo, escolha Amazon URL S3.
4. Digite o seguinte URL e escolha Avançar.

```
https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
```

5. Insira um nome para a pilha, (por exemplo, **`qldb-sample-app`**), e escolha Próximo.
6. Você pode adicionar qualquer tag conforme adequado e manter as opções padrão. Em seguida, escolha Próximo.
7. Revise as configurações de pilha e escolha Criar pilha. O AWS CloudFormation script pode levar alguns minutos para ser concluído.

   Esse script provisiona seu AWS Cloud9 ambiente com uma instância associada do Amazon Elastic Compute Cloud (AmazonEC2) que você usa para executar o aplicativo de QLDB amostra neste tutorial. Ele também clona o repositório aws-samples/ amazon-qldb-dmv-sample -java de seu ambiente de desenvolvimento. GitHub AWS Cloud9

Parte 2: Configure seu IDE

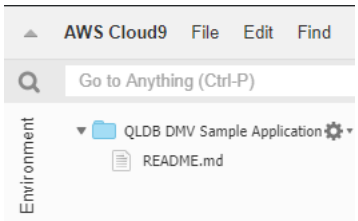
Nesta etapa, você concluirá a configuração do seu ambiente de desenvolvimento de nuvem. Você baixa e executa um script de shell fornecido para configurá-lo AWS Cloud9 IDE com as dependências do aplicativo de amostra.

Para configurar seu AWS Cloud9 ambiente

1. Abra o AWS Cloud9 console em https://console.aws.amazon.com/cloud9/.
2. Em Seus ambientes, localize a placa para o ambiente chamado Aplicativo de QLDB DMV amostra e escolha Abrir IDE. Seu ambiente pode levar um minuto para carregar quando a EC2 instância subjacente é iniciada.

   Seu AWS Cloud9 ambiente está pré-configurado com as dependências do sistema necessárias para executar o tutorial. No painel de navegação Ambiente do seu console, confirme se você vê

uma pasta chamada `QLDB DMV Sample Application`. A captura de tela a seguir do AWS Cloud9 console mostra o painel de pastas do ambiente QLDB DMV Sample Application.



Se você não vê um painel de navegação, alterne a guia Ambiente no lado esquerdo do console. Se você não vir nenhuma pasta no painel, ative Mostrar raiz do ambiente usando o ícone de configurações (⚙).

3. No painel inferior do console, você deve ver uma janela de terminal `bash` aberta. Se você não vir isso, escolha Novo terminal no menu Janela na parte superior do console.
4. Em seguida, baixe e execute um script de configuração para instalar o Open JDK 8 e, se aplicável, confira a ramificação apropriada no repositório Git. No AWS Cloud9 terminal que você criou na etapa anterior, execute os dois comandos a seguir na ordem:

   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
   ```

   Após a conclusão, você deverá ver a seguinte mensagem impressa no terminal:

   ```
   ** DMV Sample App setup completed , enjoy!! **
   ```

5. Reserve um momento para pesquisar o código do aplicativo de amostra em AWS Cloud9, particularmente no seguinte caminho de diretório:`src/main/java/software/amazon/qldb/tutorial`.

Parte 3: Executar o aplicativo QLDB DMV de amostra

Nesta etapa, você aprenderá a executar as tarefas de QLDB DMV exemplo do aplicativo Amazon usando AWS Cloud9. Para executar o código de amostra, volte ao seu AWS Cloud9 terminal ou crie uma nova janela de terminal, como você fez na Parte 2: Configurar seu IDE.

Como executar o aplicativo de amostra

1. Execute o seguinte comando em seu terminal para alternar para o diretório raiz do projeto:

```
cd ~/environment/amazon-qldb-dmv-sample-java
```

   Verifique se você está executando os exemplos no caminho de diretório a seguir.

```
/home/ec2-user/environment/amazon-qldb-dmv-sample-java/
```

2. O comando a seguir mostra a sintaxe do Gradle para executar cada tarefa.

```
./gradlew run -Dtutorial=Task
```

   Por exemplo, execute o comando a seguir para listar todos os livros contábeis em sua região Conta da AWS e na atual.

```
./gradlew run -Dtutorial=ListLedgers
```

3. Continue para Etapa 1: criar um novo ledger para iniciar o tutorial e criar um ledger.
4. (Opcional) Depois de concluir o tutorial, limpe seus recursos AWS CloudFormation caso não precise mais deles.

   a. Abra o AWS CloudFormation console em https://console.aws.amazon.com/cloudformation e exclua a pilha que você criou na Parte 1: Provisionar seus recursos.

   b. Exclua também a AWS Cloud9 pilha que o AWS CloudFormation modelo criou para você.

Etapa 1: criar um novo ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você cria um novo QLDB livro contábil da Amazon chamado`vehicle-registration`.

Para criar um novo ledger

1. Examine o arquivo a seguir (`Constants.java`), que contém valores constantes usados por todos os outros programas deste tutorial.

   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**
>
> Para o pacote Amazon Ion, você deve usar o namespace `com.amazon.ion` em seu aplicativo. AWS SDK for Java Depende de outro pacote Ion no namespace`software.amazon.ion`, mas esse é um pacote legado que não é compatível com o QLDB driver.

```
/*
 * 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**
>
> Essa classe `Constants` inclui uma instância da classe Jackson de código aberto `IonValueMapper`. Você pode usar esse mapeador para processar seus dados do Amazon Ion ao fazer transações de leitura e gravação.

O arquivo `CreateLedger.java` também tem uma dependência do seguinte programa (`DescribeLedger.java`), que descreve o status atual do seu ledger.

```
/*
 * 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. Compile e execute o programa `CreateLedger.java` para criar um ledger chamado `vehicle-registration`.

   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**
>
> - Na chamada `createLedger`, você deve especificar um nome de ledger e um modo de permissões. Recomendamos o uso do modo de permissões `STANDARD` para maximizar a segurança dos dados do seu ledger.
> - Ao criar um ledger, a proteção contra exclusão é habilitada, por padrão. Esse é um recurso QLDB que impede que os livros contábeis sejam excluídos por qualquer usuário. Você tem a opção de desativar a proteção contra exclusão na criação do livro contábil usando o QLDB API ou o AWS Command Line Interface ().AWS CLI
> - Se preferir, especifique também tags para anexar ao ledger.

Para verificar sua conexão com o novo ledger, vá para Etapa 2: Testar a conectividade com o ledger.

Etapa 2: Testar a conectividade com o ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você verifica se pode se conectar ao `vehicle-registration` livro contábil na Amazon QLDB usando o endpoint de dados transacionais. API

Para testar a conexão com o ledger

1. Use o programa (`ConnectToLedger.java`) a seguir para criar uma conexão de sessão de dados com o ledger `vehicle-registration`.

   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**
>
> - Para executar operações de dados em seu ledger, você deve criar uma instância da classe `QldbDriver` para se conectar a um ledger específico. Esse é um objeto cliente diferente do cliente `AmazonQLDB` que você usou na etapa anterior para criar o ledger. Esse cliente anterior é usado somente para executar as API operações de gerenciamento listadas noQLDBAPIReferência da Amazon.
> - Primeiro, crie um objeto `QldbDriver`. Você deve especificar um nome de ledger ao criar esse objeto de driver.
>
> Em seguida, você pode usar o método `execute` desse driver para executar instruções partiQL.
> - Opcionalmente, você pode especificar um número máximo de tentativas de repetição para exceções de transação. O `execute` método repete

automaticamente conflitos otimistas de controle de concorrência (OCC) e outras exceções transitórias comuns até esse limite configurável. O valor padrão é 4.

Se a transação ainda falhar após o limite ser atingido, o driver lançará a exceção. Para saber mais, consulte Entendendo a política de repetição com o motorista na 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

- Para executar operações de dados em seu ledger, você deve criar uma instância da classe `PooledQldbDriver` ou `QldbDriver` para se conectar a um ledger específico. Esse é um objeto cliente diferente do cliente `AmazonQLDB` que você usou na etapa anterior para criar o ledger. Esse cliente anterior é usado somente

para executar as API operações de gerenciamento listadas noQLDBAPIReferência da Amazon.

Recomendamos usar `PooledQldbDriver`, a menos que você precise implementar um pool de sessões personalizado com `QldbDriver`. O tamanho padrão do pool `PooledQldbDriver` é o número máximo de HTTP conexões abertas que o cliente da sessão permite.

- Primeiro, crie um objeto `PooledQldbDriver`. Você deve especificar um nome de ledger ao criar esse objeto de driver.

  Em seguida, você pode usar o método `execute` desse driver para executar instruções partiQL. Ou você pode criar manualmente uma sessão a partir desse objeto de driver agrupado e usar o método `execute` da sessão. Uma sessão representa uma única conexão com o ledger.

- Opcionalmente, você pode especificar um número máximo de tentativas de repetição para exceções de transação. O `execute` método repete automaticamente conflitos otimistas de controle de concorrência (OCC) e outras exceções transitórias comuns até esse limite configurável. O valor padrão é 4.

  Se a transação ainda falhar após o limite ser atingido, o driver lançará a exceção. Para saber mais, consulte Entendendo a política de repetição com o motorista na Amazon QLDB.

2. Compile e execute o programa `ConnectToLedger.java` para testar a conexão de sessão de dados com o ledger `vehicle-registration`.

Anulando o Região da AWS

O aplicativo de amostra se conecta ao QLDB seu padrão Região da AWS, que você pode definir conforme descrito na etapa de pré-requisito. Como credenciar suas credenciais AWS e região padrão Você também pode alterar a região modificando as propriedades do construtor do cliente de QLDB sessão.

2.x

O exemplo de código a seguir instancia um novo objeto `QldbSessionClientBuilder`.

```
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);
```

Você pode usar o `region` método para executar seu código QLDB em qualquer região em que ele esteja disponível. Para obter uma lista completa, consulte os QLDBendpoints e cotas da Amazon no. Referência geral da AWS

1.x

O exemplo de código a seguir instancia um novo objeto `AmazonQLDBSessionClientBuilder`.

```
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);
```

Você pode usar o `withRegion` método para executar seu código QLDB em qualquer região em que ele esteja disponível. Para obter uma lista completa, consulte os QLDBendpoints e cotas da Amazon no. Referência geral da AWS

Para criar tabelas no ledger `vehicle-registration`, vá para Etapa 3: criar tabelas, índices e dados de amostra.

Etapa 3: criar tabelas, índices e dados de amostra

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Quando seu QLDB livro contábil da Amazon está ativo e aceita conexões, você pode começar a criar tabelas para dados sobre veículos, seus proprietários e suas informações de registro. Depois de criar as tabelas e os índices, você pode carregá-los com dados.

Nesta etapa, você cria quatro tabelas no ledger `vehicle-registration`:

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

Você também cria os índices a seguir.

Nome da tabela	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Ao inserir dados de amostra, primeiro você insere documentos na tabela `Person`. Em seguida, você usa o sistema atribuído `id` a partir de cada documento `Person` para preencher os campos correspondentes nos documentos `VehicleRegistration` e `DriversLicense` apropriados.

> **Tip**
>
> Como prática recomendada, use o `id` de um documento atribuído pelo sistema como uma chave estrangeira. Embora você possa definir campos destinados a serem identificadores exclusivos (por exemplo, de um veículoVIN), o verdadeiro identificador exclusivo de um documento é seu`id`. Esse campo está incluído nos metadados do documento, que você pode consultar na visualização confirmada (a visualização definida pelo sistema de uma tabela).
>
> Para obter mais informações sobre exibições emQLDB, consulteConceitos principais. Para saber mais sobre metadados, consulte Consultar metadados do documento.

Como configurar os dados de exemplo

1. Revise os seguintes arquivos `.java`: Essas classes de modelo representam documentos que você armazena nas tabelas `vehicle-registration`. Eles são serializáveis de e para o formato Amazon Ion.

> **Note**
>
> QLDBDocumentos da Amazonsão armazenados no formato Ion, que é um superconjunto deJSON. Portanto, você pode usar a biblioteca Faster XML Jackson para modelar os dadosJSON.

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. Examine o arquivo (`SampleData.java`) a seguir, que representa os dados de amostra que você insere nas tabelas `vehicle-registration`.

   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**
>
> Para o pacote Amazon Ion, você deve usar o namespace `com.amazon.ion` em seu aplicativo. AWS SDK for Java Depende de outro pacote Ion no namespace`software.amazon.ion`, mas esse é um pacote legado que não é compatível com o QLDB driver.

```
/*
 * 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**
>
> - Essa classe usa bibliotecas Ion para fornecer métodos auxiliares que convertem seus dados de e para o formato Ion.
> - O método `getDocumentId` executa uma consulta em uma tabela com o prefixo `_ql_committed_`. Esse é um prefixo reservado que significa que você deseja consultar a visualização confirmada de uma tabela. Nessa exibição, seus dados estão aninhados no campo `data` e os metadados estão aninhados no campo `metadata`.

3. Compile e execute o programa (`CreateTable.java`) a seguir para criar as tabelas mencionadas anteriormente.

   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**
>
> Este programa demonstra como passar um lambda `TransactionExecutor` para o método `execute`. Neste exemplo, você executa várias instruções PartiQL `CREATE TABLE` em uma única transação usando uma expressão lambda.
> Esse método `execute` inicia implicitamente uma transação, executa todas as instruções no lambda e, em seguida, confirma automaticamente a transação.

4. Compile e execute o programa (`CreateIndex.java`) a seguir para criar índices nas tabelas, conforme descrito anteriormente.

   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. Compile e execute o programa a seguir (`InsertDocument.java`) para inserir os dados de exemplo em suas tabelas.

   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

- Este programa demonstra como chamar o método `execute` com valores parametrizados. Você pode passar parâmetros de tipo `IonValue` além da instrução partiQL que deseja executar. Use um ponto de interrogação (?) como um marcador variável em sua string de instrução.

- Se uma instrução `INSERT` for bem-sucedida, ela retornará a `id` de cada documento inserido.

Em seguida, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`. Vá para Etapa 4: consultar as tabelas em um ledger.

Etapa 4: consultar as tabelas em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de criar tabelas em um QLDB livro contábil da Amazon e carregá-las com dados, você pode executar consultas para revisar os dados de registro do veículo que você acabou de inserir. QLDBusa o PartiQL como sua linguagem de consulta e o Amazon Ion como seu modelo de dados orientado a documentos.

O partiQL é uma linguagem de consulta de código aberto SQL compatível que foi estendida para funcionar com o Ion. Com o partiQL, você pode inserir, consultar e gerenciar seus dados com operadores conhecidosSQL. Amazon Ion é um superconjunto de. JSON O Ion é um formato de dados de código aberto baseado em documentos que oferece a flexibilidade de armazenar e processar dados estruturados, semiestruturados e aninhados.

Nesta etapa, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`.

> **Warning**
>
> Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. O partiQL suporta essas consultas porque é compatível. SQL No entanto, não execute varreduras de tabela para casos de uso de produção emQLDB. Verificações de tabela podem causar problemas de performance em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID

de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Para ter mais informações, consulte Otimizar a performance da consulta.

Para consultar as tabelas

- Compile e execute o seguinte programa (`FindVehicles.java`) para consultar todos os veículos registrados sob uma pessoa em seu ledger.

  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**
>
> Primeiro, esse programa consulta a tabela `Person` para o documento `GovId` `LEWISR261LL` para obter seu campo de metadados `id`.
> Em seguida, usa esse documento `id` como uma chave externa para consultar a tabela `VehicleRegistration`, `PrimaryOwner.PersonId`. Ele também junta `VehicleRegistration` à tabela `Vehicle` no campo `VIN`.

Para saber mais sobre a modificação de documentos nas tabelas do ledger `vehicle-registration`, consulte Etapa 5: Modificar documentos em um ledger.

Etapa 5: Modificar documentos em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Agora que você tem dados com os quais trabalhar, pode começar a fazer alterações nos documentos do `vehicle-registration` livro contábil na AmazonQLDB. Nesta etapa, os exemplos de código a seguir demonstram como executar instruções de linguagem de manipulação de dados (DML). Essas declarações atualizam o proprietário principal de um veículo e adicionam um proprietário secundário a outro veículo.

Para modificar documentos

1. Compile e execute o seguinte programa (`TransferVehicleOwnership.java`) para atualizar o proprietário principal do veículo VIN `1N4AL11D75C109151` em seu livro contábil.

   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. Compile e execute o seguinte programa (`AddSecondaryOwner.java`) para adicionar um proprietário secundário ao veículo que está VIN `KM8SRDHF6EU074761` em seu livro contábil.

   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.");
    }
}
```

Para revisar essas alterações no ledger `vehicle-registration`, consulte [Etapa 6: Visualizar o histórico de revisão de um documento](#).

Etapa 6: Visualizar o histórico de revisão de um documento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de modificar os dados de registro para um veículo na etapa anterior, você pode consultar o histórico de todos os proprietários registrados e quaisquer outros campos atualizados. Nesta etapa, você consulta o histórico de revisão de um documento na tabela `VehicleRegistration` do ledger `vehicle-registration`.

Visualizar o histórico de revisão

1. Analise o programa a seguir (`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**
>
> - Você pode visualizar o histórico de revisões de um documento consultando a sintaxe Função de histórico incorporada a seguir.
>
> ```
> SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
> [ WHERE h.metadata.id = 'id' ]
> ```
>
> - A hora de início e a hora de término são opcionais. São valores literais do Amazon Ion que podem ser indicados com acentos graves (`` `...` ``). Para saber mais, consulte Consultando o Ion com o PartiQL na Amazon QLDB.
> - Como prática recomendada, qualifique uma consulta de histórico com um intervalo de datas (hora de início e hora de término) e um ID de documentos (`metadata.id`). QLDBprocessa `SELECT` consultas em transações, que estão sujeitas a um limite de tempo limite de transação.
>
> QLDBo histórico é indexado pela ID do documento, e você não pode criar índices de histórico adicionais no momento. As consultas de histórico que incluem uma hora de início e uma hora de término ganham o benefício da qualificação por intervalo de datas.

2. Compile e execute o `QueryHistory.java` programa para consultar o histórico de revisão do `VehicleRegistration` documento com `VIN1N4AL11D75C109151`.

Para verificar uma revisão de documento criptograficamente no ledger `vehicle-registration`, vá para Etapa 7: verificar um documento em um ledger.

Etapa 7: verificar um documento em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Com a AmazonQLDB, você pode verificar com eficiência a integridade de um documento no diário do seu livro contábil usando hash criptográfico com -256. SHA Para saber mais sobre como a verificação e o hashing criptográfico funcionam emQLDB, consulte. Verificação de dados na Amazon QLDB

Nesta etapa, você verifica uma revisão do documento na tabela `VehicleRegistration` do seu ledger `vehicle-registration`. Primeiro, você solicita um resumo, que é retornado como um arquivo de saída e atua como uma assinatura de todo o histórico de alterações do seu ledger. Em seguida, você solicita uma prova da revisão em relação a esse resumo. Usando essa prova, a integridade da sua revisão é verificada se todas as verificações de validação forem aprovadas.

Para verificar a revisão de um documento

1. Examine os `.java` arquivos a seguir, que representam QLDB objetos necessários para verificação e classes utilitárias com métodos auxiliares para valores de íon e string.

   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. Use dois arquivos `.java` (`GetDigest.java` e `GetRevision.java`) para realizar as seguintes etapas:

   - Solicite um novo resumo do ledger `vehicle-registration`.
   - Solicite uma prova para cada revisão de um documento da tabela `VehicleRegistration`.
   - Verifique as revisões usando o resumo e a prova retornados, recalculando o resumo.

   O programa `GetDigest.java` contém o código a seguir.

```
/*
 * 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**
>
> Use o método `getDigest` para solicitar um resumo que cubra a ponta atual do diário em seu ledger. A dica do diário se refere ao último bloco confirmado no momento em que QLDB recebe sua solicitação.

O programa `GetRevision.java` contém o código a seguir.

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**
>
> Depois que o `getRevision` método retorna uma prova da revisão do documento especificada, esse programa usa um lado do cliente API para verificar essa revisão. Para obter uma visão geral do algoritmo usado por issoAPI, consulteComo usar uma prova para recalcular seu resumo.

3. Compile e execute o `GetRevision.java` programa para verificar criptograficamente o `VehicleRegistration` documento com. VIN `1N4AL11D75C109151`

Para exportar e validar os dados do diário no ledger `vehicle-registration`, vá para Etapa 8: exportar e validar dados do diário em um ledger.

Etapa 8: exportar e validar dados do diário em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, você pode acessar o conteúdo do diário em seu livro contábil para vários fins, como retenção de dados, análise e auditoria. Para ter mais informações, consulte Exportação de dados de periódicos da Amazon QLDB.

Nesta etapa, você exporta blocos de diário do ledger `vehicle-registration` para um bucket do Amazon S3. Em seguida, você usa os dados exportados para validar a cadeia de hash entre os blocos de diário e os componentes de hash individuais dentro de cada bloco.

A entidade principal AWS Identity and Access Management (IAM) que você usa deve ter IAM permissões suficientes para criar um bucket do Amazon S3 em seu. Conta da AWS Para obter mais informações, consulte Políticas e permissões no Amazon S3 no Guia do usuário do Amazon S3. Você também deve ter permissões para criar uma IAM função com uma política de permissões anexada que permita QLDB gravar objetos em seu bucket do Amazon S3. Para saber mais, consulte Permissões necessárias para acessar IAM recursos no Guia do IAM usuário.

Exportar e validar dados do diário

1. Examine o arquivo a seguir (`JournalBlock.java`), que representa um bloco de diário e seu conteúdo de dados. Ele inclui um método chamado `verifyBlockHash()` que demonstra como calcular cada componente individual de um hash do bloco.

```
/*
 * 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. Compile e execute o seguinte programa (`ValidateQldbHashChain.java`) para executar as seguintes etapas:

   1. Exporte blocos de diário do `vehicle-registration` livro contábil para um bucket do Amazon S3 **`qldb-tutorial-journal-export-`*`111122223333`*** chamado (substitua pelo Conta da AWS seu número).
   2. Valide os componentes de hash individuais em cada bloco chamando `verifyBlockHash()`.
   3. Valide a cadeia de hash entre os blocos de diário.

```
/*
 * 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;
        }
    }

}
```

Se você não precisar mais usar o ledger `vehicle-registration`, prossiga para Etapa 9 (opcional): Limpar os recursos.

Etapa 9 (opcional): Limpar os recursos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode continuar usando o ledger `vehicle-registration`. No entanto, se não precisar mais dele, deverá excluí-lo.

Para excluir o ledger

1. Compile e execute o programa (`DeleteLedger.java`) a seguir para excluir seu ledger `vehicle-registration` e todo o seu conteúdo.

```
/*
 * 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**
>
> Se a proteção contra exclusão estiver ativada para seu livro contábil, você deverá primeiro desativá-la antes de excluir o livro contábil usando o. QLDB API

2. Se você exportou dados do diário na etapa anterior e não precisa mais deles, use o console do Amazon S3 para excluir seu bucket do S3.

   Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/

Tutorial QLDB do Amazon Node.js

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta implementação do aplicativo de amostra do tutorial, você usa o QLDB driver da Amazon com o AWS SDK for JavaScript em Node.js para criar um QLDB livro contábil e preenchê-lo com dados de amostra.

Ao trabalhar neste tutorial, você pode consultar a AWS SDK for JavaScript APIReferência para API operações de gerenciamento. Para operações de dados transacionais, você pode consultar a referência do QLDBdriver para Node.js API.

> **Note**
>
> Quando aplicável, algumas etapas do tutorial têm comandos ou exemplos de código diferentes para cada versão principal compatível do QLDB driver para Node.js.

Tópicos

- Instalando o aplicativo de amostra Amazon QLDB Node.js
- Etapa 1: criar um novo ledger
- Etapa 2: Testar a conectividade com o ledger
- Etapa 3: criar tabelas, índices e dados de amostra
- Etapa 4: consultar as tabelas em um ledger
- Etapa 5: Modificar documentos em um ledger
- Etapa 6: Visualizar o histórico de revisão de um documento
- Etapa 7: verificar um documento em um ledger
- Etapa 8 (opcional): Limpar os recursos

Instalando o aplicativo de amostra Amazon QLDB Node.js

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve como instalar e executar o aplicativo de QLDB amostra fornecido pela Amazon para o tutorial step-by-step Node.js. O caso de uso desse exemplo de aplicativo é um banco de dados do departamento de veículos motorizados (DMV) que rastreia as informações históricas completas sobre registros de veículos.

O aplicativo DMV de amostra para Node.js é de código aberto no GitHub repositório amazon-qldb-dmv-sampleaws-samples/ -nodejs.

Pré-requisitos

Antes de começar, certifique-se de concluir o QLDB driver para Node.jsPré-requisitos. Isso inclui instalar o Node.js e fazer o seguinte:

1. Inscreva-se em AWS.
2. Crie um usuário com as QLDB permissões apropriadas.
3. Conceda acesso programático para desenvolvimento.

Para concluir todas as etapas deste tutorial, você precisa de acesso administrativo total ao seu recurso contábil por meio do QLDBAPI.

Instalação

Para instalar o aplicativo de exemplo.

1. Digite o comando a seguir para clonar o aplicativo de GitHub amostra.

   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
   ```

   O aplicativo de amostra empacota o código-fonte completo deste tutorial e suas dependências, incluindo o driver Node.js e o AWS SDKfor JavaScript em Node.js. Este aplicativo está escrito em TypeScript.

2. Mude para o diretório em que o pacote `amazon-qldb-dmv-sample-nodejs` foi clonado.

   ```
   cd amazon-qldb-dmv-sample-nodejs
   ```

3. Faça uma instalação limpa das dependências.

   ```
   npm ci
   ```

4. Transpile o pacote.

   ```
   npm run build
   ```

   Os JavaScript arquivos transpilados são gravados no `./dist` diretório.

5. Continue para Etapa 1: criar um novo ledger para iniciar o tutorial e criar um ledger.

Etapa 1: criar um novo ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você cria um novo QLDB livro contábil da Amazon chamado`vehicle-registration`.

Para criar um novo ledger

1. Examine o arquivo a seguir (`Constants.ts`), que contém valores constantes usados por todos os outros programas deste tutorial.

```
/*
 * 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. Use o programa a seguir (`CreateLedger.ts`) para criar um ledger chamado `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 { 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**
>
> - Na chamada `createLedger`, você deve especificar um nome de ledger e um modo de permissões. Recomendamos o uso do modo de permissões `STANDARD` para maximizar a segurança dos dados do seu ledger.
> - Ao criar um ledger, a proteção contra exclusão é habilitada, por padrão. Esse é um recurso QLDB que impede que os livros contábeis sejam excluídos por qualquer usuário. Você tem a opção de desativar a proteção contra exclusão na criação do livro contábil usando o QLDB API ou o AWS Command Line Interface ().AWS CLI
> - Se preferir, especifique também tags para anexar ao ledger.

3. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/CreateLedger.js
```

Para verificar sua conexão com o novo ledger, vá para Etapa 2: Testar a conectividade com o ledger.

Etapa 2: Testar a conectividade com o ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você verifica se pode se conectar ao `vehicle-registration` livro contábil na Amazon QLDB usando o endpoint de dados transacionais. API

Para testar a conexão com o ledger

1. Use o programa a seguir (`ConnectToLedger.ts`) para criar uma conexão de sessão de dados com o ledger `vehicle-registration`.

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**
>
> Para executar transações de dados em seu livro contábil, você deve criar um objeto de QLDB driver para se conectar a um livro contábil especificado. Esse é um objeto cliente diferente do objeto `qldbClient` que você usou na etapa anterior para criar o ledger. Esse cliente anterior é usado somente para executar as API operações de gerenciamento listadas noQLDBAPIReferência da Amazon.

2. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/ConnectToLedger.js
```

Para criar tabelas no ledger `vehicle-registration`, vá para Etapa 3: criar tabelas, índices e dados de amostra.

Etapa 3: criar tabelas, índices e dados de amostra

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Quando seu QLDB livro contábil da Amazon está ativo e aceita conexões, você pode começar a criar tabelas para dados sobre veículos, seus proprietários e suas informações de registro. Depois de criar as tabelas e os índices, você pode carregá-los com dados.

Nesta etapa, você cria quatro tabelas no ledger `vehicle-registration`:

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

Você também cria os índices a seguir.

Nome da tabela	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Ao inserir dados de amostra, primeiro você insere documentos na tabela `Person`. Em seguida, você usa o sistema atribuído `id` a partir de cada documento `Person` para preencher os campos correspondentes nos documentos `VehicleRegistration` e `DriversLicense` apropriados.

> **Tip**
>
> Como prática recomendada, use o `id` de um documento atribuído pelo sistema como uma chave estrangeira. Embora você possa definir campos destinados a serem identificadores exclusivos (por exemplo, de um veículoVIN), o verdadeiro identificador exclusivo de um documento é o deleid. Esse campo está incluído nos metadados do documento, que você pode consultar na visualização confirmada (a visualização definida pelo sistema de uma tabela).
> Para obter mais informações sobre visualizações emQLDB, consulteConceitos principais.
> Para saber mais sobre metadados, consulte Consultar metadados do documento.

Para criar tabelas e índices

1. Use o programa a seguir (`CreateTable.ts`) para criar as tabelas mencionadas anteriormente.

```
/*
 * 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**
>
> Este programa demonstra como usar a `executeLambda` função em uma instância do QLDB driver. Neste exemplo, você executa várias instruções PartiQL `CREATE TABLE` com uma única expressão lambda.
> Essa função de execução inicia implicitamente uma transação, executa todas as instruções no lambda e, em seguida, confirma automaticamente a transação.

2. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/CreateTable.js
```

3. Use o programa a seguir (`CreateIndex.ts`) para criar índices nas tabelas, conforme descrito anteriormente.

```
/*
 * 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. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/CreateIndex.js
```

Para carregar os dados de exemplo em tabelas

1. Revise os seguintes arquivos `.ts`:

    1. `SampleData.ts`: contém os dados de amostra que você insere nas tabelas `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`: um módulo utilitário que importa do pacote `ion-js` para fornecer funções auxiliares que convertem, analisam e imprimem dados do Amazon Ion.

```
/*
 * 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**
>
> A `getDocumentId` função executa uma consulta que retorna um documento atribuído pelo sistema IDs de uma tabela. Para saber mais, consulte Usar a cláusula BY para consultar o ID do documento.

2. Use o programa a seguir (`InsertDocument.ts`) para inserir os dados de amostra em suas tabelas.

```
/*
 * 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

- Este programa demonstra como chamar a função `execute` com valores parametrizados. Você pode passar parâmetros de dados além da instrução partiQL

que deseja executar. Use um ponto de interrogação (?) como um marcador variável em sua string de instrução.

- Se uma instrução `INSERT` for bem-sucedida, ela retornará a `id` de cada documento inserido.

3. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/InsertDocument.js
```

Em seguida, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`. Vá para Etapa 4: consultar as tabelas em um ledger.

Etapa 4: consultar as tabelas em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de criar tabelas em um QLDB livro contábil da Amazon e carregá-las com dados, você pode executar consultas para revisar os dados de registro do veículo que você acabou de inserir. QLDBusa o PartiQL como sua linguagem de consulta e o Amazon Ion como seu modelo de dados orientado a documentos.

O partiQL é uma linguagem de consulta de código aberto SQL compatível que foi estendida para funcionar com o Ion. Com o partiQL, você pode inserir, consultar e gerenciar seus dados com operadores conhecidosSQL. Amazon Ion é um superconjunto de. JSON O Ion é um formato de dados de código aberto baseado em documentos que oferece a flexibilidade de armazenar e processar dados estruturados, semiestruturados e aninhados.

Nesta etapa, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`.

> **Warning**
>
> Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. O partiQL suporta essas consultas porque é compatível. SQL No entanto, não execute varreduras de tabela para casos de uso de produção emQLDB. Verificações de tabela podem causar problemas de performance em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Para ter mais informações, consulte Otimizar a performance da consulta.

Para consultar as tabelas

1. Use o seguinte programa (`FindVehicles.ts`) para consultar todos os veículos registrados sob uma pessoa em seu ledger.

```
/*
 * 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**
>
> Primeiro, esse programa consulta a tabela `Person` para o documento `GovId` `LEWISR261LL` para obter seu campo de metadados `id`.
> Em seguida, usa esse documento `id` como uma chave externa para consultar a tabela `VehicleRegistration`, `PrimaryOwner.PersonId`. Ele também junta `VehicleRegistration` à tabela `Vehicle` no campo `VIN`.

2. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/FindVehicles.js
```

Para saber mais sobre a modificação de documentos nas tabelas do ledger `vehicle-registration`, consulte Etapa 5: Modificar documentos em um ledger.

Etapa 5: Modificar documentos em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Agora que você tem dados com os quais trabalhar, pode começar a fazer alterações nos documentos do `vehicle-registration` livro contábil na AmazonQLDB. Nesta etapa, os exemplos de código a seguir demonstram como executar instruções de linguagem de manipulação

de dados (DML). Essas declarações atualizam o proprietário principal de um veículo e adicionam um proprietário secundário a outro veículo.

Para modificar documentos

1. Use o programa a seguir (`TransferVehicleOwnership.ts`) para atualizar o proprietário principal do veículo VIN `1N4AL11D75C109151` em seu livro contábil.

```
/*
 * 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. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/TransferVehicleOwnership.js
```

3. Use o programa a seguir (`AddSecondaryOwner.ts`) para adicionar um proprietário secundário ao veículo VIN `KM8SRDHF6EU074761` em seu livro contábil.

```
/*
 * 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. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/AddSecondaryOwner.js
```

Para revisar essas alterações no ledger `vehicle-registration`, consulte Etapa 6: Visualizar o histórico de revisão de um documento.

Etapa 6: Visualizar o histórico de revisão de um documento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de modificar os dados de registro para um veículo na etapa anterior, você pode consultar o histórico de todos os proprietários registrados e quaisquer outros campos atualizados. Nesta etapa, você consulta o histórico de revisão de um documento na tabela `VehicleRegistration` do ledger `vehicle-registration`.

Visualizar o histórico de revisão

1. Use o seguinte programa (`QueryHistory.ts`) para consultar o histórico de revisão do `VehicleRegistration` documento com `VIN1N4AL11D75C109151`.

```
/*
 * 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**
>
> - Você pode visualizar o histórico de revisões de um documento consultando a sintaxe Função de histórico incorporada a seguir.
>
> ```
> SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
> [ WHERE h.metadata.id = 'id' ]
> ```
>
> - A hora de início e a hora de término são opcionais. São valores literais do Amazon Ion que podem ser indicados com acentos graves (`` `...` ``). Para ter mais informações, consulte Consultando o Ion com o PartiQL na Amazon QLDB.
> - Como prática recomendada, qualifique uma consulta de histórico com um intervalo de datas (hora de início e hora de término) e um ID de documentos (`metadata.id`). QLDBprocessa `SELECT` consultas em transações, que estão sujeitas a um limite de tempo limite de transação.
>
> QLDBo histórico é indexado pela ID do documento e você não pode criar índices de histórico adicionais no momento. As consultas de histórico que incluem uma hora de início e uma hora de término ganham o benefício da qualificação por intervalo de datas.

2. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/QueryHistory.js
```

Para verificar uma revisão de documento criptograficamente no ledger `vehicle-registration`, vá para Etapa 7: verificar um documento em um ledger.

Etapa 7: verificar um documento em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Com a AmazonQLDB, você pode verificar com eficiência a integridade de um documento no diário do seu livro contábil usando hash criptográfico com -256. SHA Para saber mais sobre como a verificação e o hashing criptográfico funcionam emQLDB, consulte. Verificação de dados na Amazon QLDB

Nesta etapa, você verifica uma revisão do documento na tabela `VehicleRegistration` do seu ledger `vehicle-registration`. Primeiro, você solicita um resumo, que é retornado como um arquivo de saída e atua como uma assinatura de todo o histórico de alterações do seu ledger. Em seguida, você solicita uma prova da revisão em relação a esse resumo. Usando essa prova, a integridade da sua revisão é verificada se todas as verificações de validação forem aprovadas.

Para verificar a revisão de um documento

1. Examine os `.ts` arquivos a seguir, que contêm os QLDB objetos necessários para verificação.

    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. Use dois programas `.ts` (`GetDigest.ts` e `GetRevision.ts`) para realizar as seguintes etapas:

   - Solicite um novo resumo do ledger `vehicle-registration`.
   - Solicite uma prova para cada revisão do documento com VIN `1N4AL11D75C109151` a `VehicleRegistration` tabela.
   - Verifique as revisões usando o resumo e a prova retornados, recalculando o resumo.

   O programa `GetDigest.ts` contém o código a seguir.

```
/*
 * 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**
>
> Use a função `getDigest` para solicitar um resumo que cubra a ponta atual do diário em seu ledger. A dica do diário se refere ao último bloco confirmado no momento em que QLDB recebe sua solicitação.

O programa `GetRevision.ts` contém o código a seguir.

```
/*
 * 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**
>
> Depois que a `getRevision` função retorna uma prova da revisão do documento especificada, esse programa usa um lado do cliente API para verificar essa revisão.

3. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/GetRevision.js
```

Se você não precisar mais usar o ledger `vehicle-registration`, prossiga para Etapa 8 (opcional): Limpar os recursos.

Etapa 8 (opcional): Limpar os recursos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode continuar usando o ledger `vehicle-registration`. No entanto, se não precisar mais dele, deverá excluí-lo.

Para excluir o ledger

1. Use o programa (`DeleteLedger.ts`) a seguir para excluir seu ledger `vehicle-registration` e todo o seu conteúdo.

```
/*
 * 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**
>
> Se a proteção contra exclusão estiver ativada para seu livro contábil, você deverá primeiro desativá-la antes de excluir o livro contábil usando o. QLDB API

2. Para executar o programa transpilado, digite o comando a seguir.

```
node dist/DeleteLedger.js
```

Tutorial do Amazon QLDB Python

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta implementação do aplicativo de amostra do tutorial, você usa o QLDB driver da Amazon com o AWS SDK for Python (Boto3) para criar um QLDB livro contábil e preenchê-lo com dados de amostra.

Ao trabalhar neste tutorial, você pode consultar o cliente de QLDB baixo nível na Referência AWS SDKpara Python (Boto3API) para operações de gerenciamento. API Para operações de dados transacionais, você pode consultar a Referência do QLDBDriver for Python. API

> **Note**
>
> Quando aplicável, algumas etapas do tutorial têm comandos ou exemplos de código diferentes para cada versão principal compatível do QLDB driver para Python.

Tópicos

- Instalação do aplicativo de QLDB amostra Amazon Python
- Etapa 1: criar um novo ledger
- Etapa 2: Testar a conectividade com o ledger
- Etapa 3: criar tabelas, índices e dados de amostra
- Etapa 4: consultar as tabelas em um ledger
- Etapa 5: Modificar documentos em um ledger
- Etapa 6: Visualizar o histórico de revisão de um documento
- Etapa 7: verificar um documento em um ledger
- Etapa 8 (opcional): Limpar os recursos

Instalação do aplicativo de QLDB amostra Amazon Python

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve como instalar e executar o aplicativo de QLDB amostra fornecido pela Amazon para o tutorial step-by-step Python. O caso de uso desse exemplo de aplicativo é um banco de dados do departamento de veículos motorizados (DMV) que rastreia as informações históricas completas sobre registros de veículos.

O aplicativo DMV de amostra para Python é de código aberto no GitHub repositório aws-samples/ -python. amazon-qldb-dmv-sample

Pré-requisitos

Antes de começar, certifique-se de concluir o QLDB driver para PythonPré-requisitos. Isso inclui instalar o Python e fazer o seguinte:

1. Inscreva-se em AWS.
2. Crie um usuário com as QLDB permissões apropriadas.
3. Conceda acesso programático para desenvolvimento.

Para concluir todas as etapas deste tutorial, você precisa de acesso administrativo total ao seu recurso contábil por meio do QLDBAPI.

Instalação

Para instalar o aplicativo de exemplo.

1. Digite o `pip` comando a seguir para clonar e instalar o aplicativo de GitHub amostra.

   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
```

O aplicativo de amostra empacota o código-fonte completo deste tutorial e suas dependências, incluindo o driver Python e o AWS SDK for Python (Boto3).

2. Antes de começar a executar o código na linha de comando, alterne seu diretório de trabalho atual para o local em que o `pyqldbsamples` pacote está instalado. Insira o comando da a seguir.

```
cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
```

3. Continue para Etapa 1: criar um novo ledger para iniciar o tutorial e criar um ledger.

Etapa 1: criar um novo ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você cria um novo QLDB livro contábil da Amazon chamado`vehicle-registration`.

Para criar um novo ledger

1. Examine o arquivo a seguir (`constants.py`), que contém valores constantes usados por todos os outros programas deste tutorial.

```
# 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. Use o programa a seguir (`create_ledger.py`) para criar um ledger chamado `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.
#
# 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**
>
> - Na chamada `create_ledger`, você deve especificar um nome de ledger e um modo de permissões. Recomendamos o uso do modo de permissões `STANDARD` para maximizar a segurança dos dados do seu ledger.
> - Ao criar um ledger, a proteção contra exclusão é habilitada, por padrão. Esse é um recurso QLDB que impede que os livros contábeis sejam excluídos por qualquer usuário. Você tem a opção de desativar a proteção contra exclusão na criação do livro contábil usando o QLDB API ou o AWS Command Line Interface ().AWS CLI
> - Se preferir, especifique também tags para anexar ao ledger.

3. Para executar o programa, digite o comando a seguir.

```
python create_ledger.py
```

Para verificar sua conexão com o novo ledger, vá para Etapa 2: Testar a conectividade com o ledger.

Etapa 2: Testar a conectividade com o ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Nesta etapa, você verifica se pode se conectar ao `vehicle-registration` livro contábil na Amazon QLDB usando o endpoint de dados transacionais. API

Para testar a conexão com o ledger

1. Use o programa a seguir (`connect_to_ledger.py`) para criar uma conexão de sessão de dados com o ledger `vehicle-registration`.

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**
>
> - Para executar transações de dados em seu livro contábil, você deve criar um objeto de QLDB driver para se conectar a um livro contábil específico. Esse é um objeto cliente diferente do objeto `qldb_client` que você usou na etapa anterior para criar o ledger. Esse cliente anterior é usado somente para executar as API operações de gerenciamento listadas noQLDBAPIReferência da Amazon.
> - Você deve especificar um nome de ledger ao criar esse objeto de driver.

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**
>
> - Para executar transações de dados em seu livro contábil, você deve criar um objeto de QLDB driver para se conectar a um livro contábil específico. Esse é um objeto cliente diferente do objeto `qldb_client` que você usou na etapa anterior para criar o ledger. Esse cliente anterior é usado somente para executar as API operações de gerenciamento listadas noQLDBAPIReferência da Amazon.
> - Primeiro, crie um objeto de QLDB driver agrupado. Você deve especificar um nome de ledger ao criar esse objeto de driver.
> - Em seguida, você pode criar sessões a partir desse objeto de driver agrupado.

2. Para executar o programa, digite o comando a seguir.

```
python connect_to_ledger.py
```

Para criar tabelas no ledger `vehicle-registration`, vá para Etapa 3: criar tabelas, índices e dados de amostra.

Etapa 3: criar tabelas, índices e dados de amostra

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Quando seu QLDB livro contábil da Amazon está ativo e aceita conexões, você pode começar a criar tabelas para dados sobre veículos, seus proprietários e suas informações de registro. Depois de criar as tabelas e os índices, você pode carregá-los com dados.

Nesta etapa, você cria quatro tabelas no ledger `vehicle-registration`:

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

Você também cria os índices a seguir.

Nome da tabela	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Ao inserir dados de amostra, primeiro você insere documentos na tabela `Person`. Em seguida, você usa o sistema atribuído `id` a partir de cada documento `Person` para preencher os campos correspondentes nos documentos `VehicleRegistration` e `DriversLicense` apropriados.

> **Tip**
>
> Como prática recomendada, use o `id` de um documento atribuído pelo sistema como uma chave estrangeira. Embora você possa definir campos destinados a serem identificadores exclusivos (por exemplo, de um veículoVIN), o verdadeiro identificador exclusivo de um documento é o deleid. Esse campo está incluído nos metadados do documento, que você pode consultar na visualização confirmada (a visualização definida pelo sistema de uma tabela).
>
> Para obter mais informações sobre exibições emQLDB, consulteConceitos principais. Para saber mais sobre metadados, consulte Consultar metadados do documento.

Para criar tabelas e índices

1. Use o programa a seguir (`create_table.py`) para criar as tabelas mencionadas anteriormente.

   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**
>
> Este programa demonstra como usar a função `execute_lambda`. Neste exemplo, você executa várias instruções PartiQL `CREATE TABLE` com uma única expressão lambda. Essa função de execução inicia implicitamente uma transação, executa todas as instruções no lambda e, em seguida, confirma automaticamente a transação.

2. Para executar o programa, digite o comando a seguir.

```
python create_table.py
```

3. Use o programa a seguir (`create_index.py`) para criar índices nas tabelas, conforme descrito anteriormente.

   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. Para executar o programa, digite o comando a seguir.

```
python create_index.py
```

Para carregar os dados de exemplo em tabelas

1. Examine o arquivo (`sample_data.py`) a seguir, que representa os dados de amostra que você insere nas tabelas `vehicle-registration`. Esse arquivo também é importado do pacote `amazon.ion` para fornecer funções auxiliares que convertem, analisam e imprimem dados do Amazon Ion.

```
# 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**
>
> A `get_document_ids` função executa uma consulta que retorna um documento atribuído pelo sistema a IDs partir de uma tabela. Para saber mais, consulte Usar a cláusula BY para consultar o ID do documento.

2. Use o programa a seguir (`insert_document.py`) para inserir os dados de amostra em suas tabelas.

   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**
>
> - Este programa demonstra como chamar a função `execute_statement` com valores parametrizados. Você pode passar parâmetros de dados além da instrução partiQL que deseja executar. Use um ponto de interrogação (?) como um marcador variável em sua string de instrução.
> - Se uma instrução `INSERT` for bem-sucedida, ela retornará a `id` de cada documento inserido.

3. Para executar o programa, digite o comando a seguir.

```
python insert_document.py
```

Em seguida, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`. Vá para Etapa 4: consultar as tabelas em um ledger.

Etapa 4: consultar as tabelas em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de criar tabelas em um QLDB livro contábil da Amazon e carregá-las com dados, você pode executar consultas para revisar os dados de registro do veículo que você acabou de inserir. QLDBusa o PartiQL como sua linguagem de consulta e o Amazon Ion como seu modelo de dados orientado a documentos.

O partiQL é uma linguagem de consulta de código aberto SQL compatível que foi estendida para funcionar com o Ion. Com o partiQL, você pode inserir, consultar e gerenciar seus dados com operadores conhecidosSQL. Amazon Ion é um superconjunto de. JSON O Ion é um formato de dados de código aberto baseado em documentos que oferece a flexibilidade de armazenar e processar dados estruturados, semiestruturados e aninhados.

Nesta etapa, você pode usar instruções `SELECT` para ler os dados das tabelas no ledger `vehicle-registration`.

> **Warning**
>
> Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. O partiQL suporta essas consultas porque é compatível. SQL No entanto, não execute varreduras de tabela para casos de uso de produção emQLDB. Verificações de tabela podem causar problemas de performance em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID

de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Para ter mais informações, consulte Otimizar a performance da consulta.

Para consultar as tabelas

1. Use o seguinte programa (`find_vehicles.py`) para consultar todos os veículos registrados sob uma pessoa em seu ledger.

   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**
>
> Primeiro, esse programa consulta a tabela `Person` para o documento `GovId` `LEWISR261LL` para obter seu campo de metadados `id`.

Em seguida, usa esse documento `id` como uma chave externa para consultar a tabela `VehicleRegistration`, `PrimaryOwner.PersonId`. Ele também junta `VehicleRegistration` à tabela `Vehicle` no campo `VIN`.

2. Para executar o programa, digite o comando a seguir.

```
python find_vehicles.py
```

Para saber mais sobre a modificação de documentos nas tabelas do ledger `vehicle-registration`, consulte Etapa 5: Modificar documentos em um ledger.

Etapa 5: Modificar documentos em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Agora que você tem dados com os quais trabalhar, pode começar a fazer alterações nos documentos do `vehicle-registration` livro contábil na AmazonQLDB. Nesta etapa, os exemplos de código a seguir demonstram como executar instruções de linguagem de manipulação de dados (DML). Essas declarações atualizam o proprietário principal de um veículo e adicionam um proprietário secundário a outro veículo.

Para modificar documentos

1. Use o programa a seguir (`transfer_vehicle_ownership.py`) para atualizar o proprietário principal do veículo VIN `1N4AL11D75C109151` em seu livro contábil.

   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. Para executar o programa, digite o comando a seguir.

```
python transfer_vehicle_ownership.py
```

3. Use o programa a seguir (`add_secondary_owner.py`) para adicionar um proprietário secundário ao veículo VIN `KM8SRDHF6EU074761` em seu livro contábil.

   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. Para executar o programa, digite o comando a seguir.

```
python add_secondary_owner.py
```

Para revisar essas alterações no ledger `vehicle-registration`, consulte Etapa 6: Visualizar o histórico de revisão de um documento.

Etapa 6: Visualizar o histórico de revisão de um documento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de modificar os dados de registro para um veículo na etapa anterior, você pode consultar o histórico de todos os proprietários registrados e quaisquer outros campos atualizados. Nesta etapa, você consulta o histórico de revisão de um documento na tabela `VehicleRegistration` do ledger `vehicle-registration`.

Visualizar o histórico de revisão

1. Use o programa a seguir (`query_history.py`) para consultar o histórico de revisão do `VehicleRegistration` documento com `VIN1N4AL11D75C109151`.

   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

- Você pode visualizar o histórico de revisões de um documento consultando a sintaxe Função de histórico incorporada a seguir.

```
SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

- A hora de início e a hora de término são opcionais. São valores literais do Amazon Ion que podem ser indicados com acentos graves (`` `...` ``). Para saber mais, consulte Consultando o Ion com o PartiQL na Amazon QLDB.
- Como prática recomendada, qualifique uma consulta de histórico com um intervalo de datas (hora de início e hora de término) e um ID de documentos (`metadata.id`).

QLDBprocessa `SELECT` consultas em transações, que estão sujeitas a um limite de tempo limite de transação.

QLDBo histórico é indexado pela ID do documento, e você não pode criar índices de histórico adicionais no momento. As consultas de histórico que incluem uma hora de início e uma hora de término ganham o benefício da qualificação por intervalo de datas.

2. Para executar o programa, digite o comando a seguir.

```
python query_history.py
```

Para verificar uma revisão de documento criptograficamente no ledger `vehicle-registration`, vá para Etapa 7: verificar um documento em um ledger.

Etapa 7: verificar um documento em um ledger

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Com a AmazonQLDB, você pode verificar com eficiência a integridade de um documento no diário do seu livro contábil usando hash criptográfico com -256. SHA Para saber mais sobre como a verificação e o hashing criptográfico funcionam emQLDB, consulte. Verificação de dados na Amazon QLDB

Nesta etapa, você verifica uma revisão do documento na tabela `VehicleRegistration` do seu ledger `vehicle-registration`. Primeiro, você solicita um resumo, que é retornado como um arquivo de saída e atua como uma assinatura de todo o histórico de alterações do seu ledger. Em seguida, você solicita uma prova da revisão em relação a esse resumo. Usando essa prova, a integridade da sua revisão é verificada se todas as verificações de validação forem aprovadas.

Para verificar a revisão de um documento

1. Examine os `.py` arquivos a seguir, que representam QLDB objetos necessários para verificação e um módulo utilitário com funções auxiliares para converter tipos de QLDB resposta em cadeias de caracteres.

   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. Use dois programas `.py` (`get_digest.py` e `get_revision.py`) para realizar as seguintes etapas:

   - Solicite um novo resumo do ledger `vehicle-registration`.
   - Solicite uma prova para cada revisão do documento com VIN `1N4AL11D75C109151` a `VehicleRegistration` tabela.
   - Verifique as revisões usando o resumo e a prova retornados, recalculando o resumo.

   O programa `get_digest.py` contém o código a seguir.

```
# 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**
>
> Use a função `get_digest_result` para solicitar um resumo que cubra a ponta atual do diário em seu ledger. A dica do diário se refere ao último bloco confirmado no momento em que QLDB recebe sua solicitação.

O programa `get_revision.py` contém o código a seguir.

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**
>
> Depois que a `get_revision` função retorna uma prova da revisão do documento especificada, esse programa usa um lado do cliente API para verificar essa revisão.

3. Para executar o programa, digite o comando a seguir.

```
python get_revision.py
```

Se você não precisar mais usar o ledger `vehicle-registration`, prossiga para Etapa 8 (opcional): Limpar os recursos.

Etapa 8 (opcional): Limpar os recursos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode continuar usando o ledger `vehicle-registration`. No entanto, se não precisar mais dele, deverá excluí-lo.

Para excluir o ledger

1. Use o programa (`delete_ledger.py`) a seguir para excluir seu ledger `vehicle-registration` e todo o seu conteúdo.

```
# 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**
>
> Se a proteção contra exclusão estiver ativada para seu livro contábil, você deverá primeiro desativá-la antes de excluir o livro contábil usando o. QLDB API

O arquivo `delete_ledger.py` também depende do seguinte programa (`describe_ledger.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 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. Para executar o programa, digite o comando a seguir.

```
python delete_ledger.py
```

Trabalhando com os tipos de dados Amazon Ion na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB armazena seus dados no formato Amazon Ion. Para trabalhar com dados emQLDB, você deve usar uma biblioteca Ion como dependência de uma linguagem de programação compatível.

Nesta seção, aprenda como converter dados de tipos nativos em seus equivalentes Ion e vice-versa. Este guia de referência mostra exemplos de código que usam o QLDB driver para processar dados de íons em um QLDB livro contábil. Inclui exemplos de código para Java,. NET(C#), Go, Node.js (TypeScript) e Python.

Tópicos

- Pré-requisitos
- Bool
- Int
- Float
- Decimal
- Timestamp
- String
- Blob
- Listar
- struct
- Valores nulos e tipos dinâmicos
- Conversão descendente para JSON

Pré-requisitos

Os exemplos de código a seguir pressupõem que você tenha uma instância de QLDB driver conectada a um livro contábil ativo com uma tabela chamada`ExampleTable`. A tabela contém um único documento existente que tem os oito campos a seguir:

- ExampleBool
- ExampleInt
- ExampleFloat
- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> Para fins dessa referência, suponha que o tipo armazenado em cada campo corresponda ao nome. Na prática, QLDB não impõe definições de esquema ou tipo de dados para campos de documentos.

Bool

Os exemplos de código a seguir mostram como processar o tipo booleano de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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

Os exemplos de código a seguir mostram como processar o tipo inteiro de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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))
```

Float

Os exemplos de código a seguir mostram como processar o tipo float de Ion.

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

Usar 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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `IResult`.

Usar C# double

```
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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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))
```

Decimal

Os exemplos de código a seguir mostram como processar o tipo decimal de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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**
>
> Você também pode usar `ionValue.numberValue()` para transformar um íon decimal em um JavaScript número. O uso de `ionValue.numberValue()` tem melhor desempenho, mas é menos preciso.

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))
```

Timestamp

Os exemplos de código a seguir mostram como processar o tipo timestamp de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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))
```

String

Os exemplos de código a seguir mostram como processar o tipo segmento de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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

Os exemplos de código a seguir mostram como processar o tipo blob de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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))
```

Listar

Os exemplos de código a seguir mostram como processar o tipo lista de Ion.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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

EmQLDB, `struct` os tipos de dados são particularmente exclusivos de outros tipos de íons. Os documentos de nível superior que você insere em uma tabela devem ser do tipo `struct`. Um campo de documento também pode armazenar um `struct` aninhado.

Para simplificar, os exemplos a seguir definem um documento que tem somente os campos `ExampleString` e `ExampleInt`.

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;
});
```

Como alternativa, você pode usar a biblioteca Jackson para mapear tipos de dados de e para o Ion. A biblioteca oferece suporte aos outros tipos de dados de Ion, mas este exemplo se concentra no tipo `struct`.

```
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

Use a Amazon. QLDB.Driver.Biblioteca de serialização para mapear tipos de dados C# nativos de e para o Ion. A biblioteca oferece suporte aos outros tipos de dados de Ion, mas este exemplo se concentra no tipo `struct`.

```
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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `IResult`.

Como alternativa, você pode usar a Amazon. IonDotnetBiblioteca .Builders para processar tipos de dados Ion.

```
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))
```

Valores nulos e tipos dinâmicos

QLDBoferece suporte a conteúdo aberto e não impõe definições de esquema ou tipo de dados para campos de documentos. Você também pode armazenar valores nulos de íons em um QLDB documento. Todos os exemplos anteriores presumem que cada tipo de dados retornado é conhecido e não é nulo. Os exemplos a seguir mostram como trabalhar com o Ion quando o tipo de dados não é conhecido ou é possivelmente nulo.

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

Limitações de marshalling

Em Go, os valores `nil` não retêm seu tipo quando você os organiza e depois os desorganiza. Um valor `nil` é agrupado para Ion null, mas Ion null desagrupa para um valor zero em vez de `nil`.

```
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!
```

Conversão descendente para JSON

Se seu aplicativo exigir JSON compatibilidade, você pode reduzir a conversão dos dados do Amazon Ion em. JSON No entanto, a JSON conversão de íon em causa perdas em certos casos em que seus dados usam os tipos ricos de íons que não existem em. JSON

Para obter detalhes sobre as regras de JSON conversão de íons em, consulte Conversão descendente em JSON no Amazon Ion Cookbook.

Trabalhando com dados e histórico na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Os tópicos a seguir fornecem exemplos básicos de instruções create, read, update e delete (CRUD). Você pode executar manualmente essas instruções usando o editor partiQL no QLDBconsole ou no QLDB shell. Este guia também explica o processo de como QLDB lidar com seus dados à medida que você faz alterações em seu livro contábil.

QLDBsuporta a linguagem de consulta partiQL.

Para exemplos de código que mostram como executar programaticamente instruções semelhantes usando o QLDB driver, consulte os tutoriais em. Conceitos básicos do driver

> **Tip**
>
> A seguir está um breve resumo das dicas e das melhores práticas para trabalhar com o PartiQL em: QLDB
>
> - Entenda os limites de simultaneidade e transação — Todas as declarações, incluindo `SELECT` consultas, estão sujeitas a conflitos otimistas de controle de simultaneidade (OCC) e limites de transação, incluindo um tempo limite de transação de 30 segundos.
> - Use índices: use índices de alta cardinalidade e execute consultas direcionadas para otimizar suas declarações e evitar a varredura completa da tabela. Para saber mais, consulte Otimizar a performance da consulta.
> - Use predicados de igualdade: as pesquisas indexadas exigem um operador de igualdade (= ou `IN`). Operadores de desigualdade (<, >, `LIKE`, `BETWEEN`) não se qualificam para pesquisas indexadas e resultam em verificações de tabela completa.
> - Use somente junções internas — QLDB suporta somente junções internas. Como prática recomendada, junte em campos indexados para cada tabela que você está unindo.

Escolha índices de alta cardinalidade para os critérios de junção e os predicados de igualdade.

Tópicos

- Criação de tabelas com índices e inserção de documentos
- Consultar seus dados
- Consultar metadados do documento
- Usar a cláusula BY para consultar o ID do documento
- Atualizar e excluir documentos
- Consultar o histórico de revisões
- Redigir revisões de documentos
- Otimizar a performance da consulta
- Obter estatísticas de instruções PartiQL
- Consultar o catálogo do sistema
- Gerenciar tabelas
- Gerenciamento de Índices
- Único IDs na Amazon QLDB

Criação de tabelas com índices e inserção de documentos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Depois de criar um QLDB livro contábil da Amazon, sua primeira etapa é criar uma tabela com uma CREATE TABLE declaração básica. As tabelas consistem em QLDBdocumentos, que são conjuntos de dados no formato Amazon Ion `struct`.

Tópicos

- Criar tabelas e índices

- Inserir documentos

Criar tabelas e índices

As tabelas têm nomes simples que diferenciam maiúsculas de minúsculas, sem namespaces. QLDBoferece suporte a conteúdo aberto e não impõe esquema, portanto, você não define atributos ou tipos de dados ao criar tabelas.

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

Uma instrução `CREATE TABLE` retorna o ID atribuído pelo sistema da nova tabela. Todos os sistemas atribuídos IDs em QLDB são identificadores universalmente exclusivos (UUID), cada um representado em uma string codificada em Base62.

> **Note**
>
> Opcionalmente, você pode definir tags para um recurso de tabela ao criar a tabela. Para saber como, consulte Marcar tabelas na criação.

Também é possível criar índices em tabelas para otimizar a performance da consulta.

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

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

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

> **Important**
>
> QLDBrequer um índice para pesquisar um documento com eficiência. Sem um índice, QLDB precisa fazer uma digitalização completa da tabela ao ler documentos. Isso pode causar problemas de desempenho em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.

Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade (= ou `IN`) em um campo indexado ou em um ID de documento. Para ter mais informações, consulte Otimizar a performance da consulta.

Observe as seguintes restrições ao criar índices:

- Um índice só pode ser criado em um único campo de nível superior. Não há suporte para índices compostos, aninhados, exclusivos e baseados em funções.
- Você pode criar um índice em qualquer tipo de dados Ion, incluindo `list` e `struct`. No entanto, você só pode fazer a pesquisa indexada pela igualdade de todo o valor de Ion, independentemente do tipo de Ion. Por exemplo, ao usar um tipo `list` como índice, você não pode fazer uma pesquisa indexada por um item dentro da lista.
- O desempenho da consulta é aprimorado somente quando você usa um predicado de igualdade; por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`.

  QLDBnão respeita as desigualdades nos predicados de consulta. Como resultado, as verificações filtradas por intervalo não são implementadas.
- Os nomes dos campos indexados diferenciam maiúsculas e minúsculas e podem ter no máximo 128 caracteres.
- A criação do índice em QLDB é assíncrona. O tempo necessário para concluir a criação de um índice em uma tabela não vazia varia dependendo do tamanho da tabela. Para ter mais informações, consulte Gerenciamento de Índices.

Inserir documentos

Depois, você pode inserir documentos em suas tabelas. QLDBos documentos são armazenados no formato Amazon Ion. As instruções INSERT partiQL a seguir incluem um subconjunto dos dados de amostra de registro do veículo usados em Começando a usar o QLDB console da Amazon.

```
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'
} >>
```

Sintaxe e semântica do PartiQL

- Os nomes dos campos estão entre aspas simples (`'...'`).

- Os valores da string também estão entre aspas simples (`'...'`).
- Os carimbos de data/hora estão entre acentos graves (`` `...` ``). Os acentos graves podem ser usados para denotar qualquer literal de Ion.
- Números inteiros e decimais são valores literais que não precisam ser indicados.

Para obter mais detalhes sobre a sintaxe e a semântica do partiQL, consulte Consultando o Ion com o PartiQL na Amazon QLDB.

Uma instrução `INSERT` cria a revisão inicial de um documento com um número de versão zero. Para identificar cada documento de forma exclusiva, QLDB atribua uma ID do documento como parte dos metadados. As instruções de inserção retornam o ID de cada documento inserido.

> **Important**
>
> Como QLDB não impõe o esquema, você pode inserir o mesmo documento em uma tabela várias vezes. Cada declaração inserida confirma uma entrada de documento separada no diário e QLDB atribui a cada documento uma ID exclusiva.

Para saber como consultar os documentos que você inseriu na tabela, vá para Consultar seus dados.

Consultar seus dados

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A visualização do usuário retorna somente a última revisão não excluída dos seus dados do usuário. Essa é a visualização padrão na AmazonQLDB. Isso significa que nenhum qualificador especial é necessário quando você deseja consultar somente seus dados.

Para obter detalhes sobre a sintaxe e os parâmetros dos exemplos de consulta a seguir, consulte a SELECT referência do Amazon QLDB PartiQL.

Tópicos

- Consultas básicas
- Projeções e filtros
- Junções
- Dados aninhados

Consultas básicas

Consultas `SELECT` básicas retornam os documentos que você inseriu na tabela.

> **Warning**
>
> Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. O partiQL suporta essas consultas porque é compatível. SQL No entanto, não execute varreduras de tabela para casos de uso de produção emQLDB. Verificações de tabela podem causar problemas de performance em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Para ter mais informações, consulte Otimizar a performance da consulta.

As consultas a seguir mostram os resultados dos documentos de registro do veículo que você inseriu anteriormente em Criação de tabelas com índices e inserção de documentos. A ordem dos resultados não é específica e pode variar para cada consulta `SELECT`. Você não deve confiar na ordem dos resultados para qualquer consulta emQLDB.

```
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**
>
> No PartiQL, você usa aspas simples para denotar cadeias de caracteres na linguagem de manipulação de dados () ou em instruções de consulta. DML Mas o QLDB console e o QLDB shell retornam os resultados da consulta no formato de texto Amazon Ion, então você vê cadeias de caracteres entre aspas duplas.
> Essa sintaxe permite que a linguagem de consulta PartiQL SQL mantenha a compatibilidade e o formato de texto Amazon Ion mantenha a JSON compatibilidade.

Projeções e filtros

Você pode fazer projeções (`SELECT` direcionadas) e outros filtros padrão (cláusulas `WHERE`). A consulta a seguir retorna um subconjunto dos campos do documento da tabela `VehicleRegistration`. Ela filtra veículos com os seguintes critérios

- Filtro de string: está registrado em Seattle.
- Filtro decimal - Tem um valor de multa pendente menor que `100.0`.
- Filtro de data: tem uma data de registro válida em ou após 4 de setembro de 2019.

```
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" }]
    }
}
```

Junções

Você também pode escrever consultas de junção internas. O exemplo a seguir mostra uma consulta de junção interna implícita que retorna todos os documentos de registro junto com os atributos dos veículos registrados.

```
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"
}
```

Ou você pode escrever a mesma consulta de junção interna na sintaxe explícita da seguinte forma.

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

Dados aninhados

Você pode usar o PartiQL in QLDB para consultar dados aninhados em documentos. O exemplo a seguir mostra uma subconsulta correlacionada que nivela os dados aninhados. O caractere `@` é tecnicamente opcional aqui. Mas isso indica explicitamente que você deseja a estrutura `Owners` em `VehicleRegistration`, não uma coleção com um nome `Owners` diferente (se houver).

```
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: []
}
```

Veja a seguir uma subconsulta na lista `SELECT` que projeta dados aninhados, além de uma junção interna.

```
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"]
}
```

A consulta a seguir retorna o número de índice `PersonId` e (ordinal) de cada pessoa na lista `Owners.SecondaryOwners` de um documento `VehicleRegistration`.

```
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
}
```

Para saber como consultar os metadados do seu documento, vá para Consultar metadados do documento.

Consultar metadados do documento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Uma instrução `INSERT` cria a revisão inicial de um documento com um número de versão zero. Para identificar cada documento de forma exclusiva, a Amazon QLDB atribui uma ID do documento como parte dos metadados.

Além do ID do documento e do número da versão, QLDB armazena outros metadados gerados pelo sistema para cada documento em uma tabela. Esses metadados incluem informações da transação, atributos do diário e o valor de hash do documento.

Todos os sistemas atribuídos IDs são identificadores universalmente exclusivos (UUID), cada um representado em uma string codificada em Base62. Para ter mais informações, consulte Único IDs na Amazon QLDB.

Tópicos

- Visão confirmada
- Unir as visualizações confirmadas e do usuário

Visão confirmada

Você pode acessar os metadados do documento consultando a visualização confirmada. Essa visualização retorna documentos DA tabela definida pelo sistema que corresponde diretamente à sua tabela de usuário. Ela inclui a última revisão confirmada e não excluída de seus dados e dos metadados gerados pelo sistema. Para consultar essa visualização, adicione o prefixo `_ql_committed_` ao nome da tabela em sua consulta. (O prefixo `_ql_` é reservado QLDB para objetos do sistema.)

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Usando os dados inseridos anteriormente em Criação de tabelas com índices e inserção de documentos, a saída dessa consulta mostra o conteúdo do sistema da revisão mais recente de cada documento não excluído. O documento do sistema tem metadados aninhados no campo `metadata` e seus dados de usuário aninhados no campo `data`.

```
{
    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"
```

```
    }
}
```

Campos de visualização confirmados

- `blockAddress`: a localização do bloco no diário do seu ledger em que a revisão do documento foi confirmada. Um endereço, que pode ser usado para verificação criptográfica, tem os dois campos a seguir.
  - `strandId`: o ID exclusivo da cadeia do diário que contém o bloco.
  - `sequenceNo`: o número do índice que especifica a localização do bloco dentro da cadeia.

> **Note**
>
> Ambos os documentos neste exemplo têm um `blockAddress` idêntico com o mesmo `sequenceNo`. Como esses documentos foram inseridos em uma única transação (e, nesse caso, em uma única declaração), eles foram confirmados no mesmo bloco.

- `hash`— O valor de hash de íons SHA -256 que representa exclusivamente a revisão do documento. O hash cobre os campos `data` e `metadata` das revisões e pode ser usado para verificação criptográfica.
- `data`: os atributos de dados do usuário do documento.

  Se você redigir uma revisão, essa estrutura `data` será substituída por um campo `dataHash`, cujo valor é o hash de Ion da estrutura `data` removida.
- `metadata`: os atributos de metadados do documento.
  - `id`: o ID exclusivo do documento atribuído pelo sistema.
  - `version`: o número da versão do documento. Esse é um número inteiro baseado em zero que é incrementado com cada revisão do documento.
  - `txTime`: o carimbo de hora/data e hora em a revisão do documento foi confirmada no diário.
  - `txId`: o ID exclusivo da transação que confirmou a revisão do documento.

Unir as visualizações confirmadas e do usuário

Você pode escrever consultas que unem uma tabela na visualização confirmada com uma tabela na visualização do usuário. Por exemplo, talvez você queira unir o documento `id` de uma tabela com um campo definido pelo usuário de outra tabela.

A consulta a seguir une duas tabelas nomeadas `DriversLicense` e `Person` em seus `PersonId` e campos `id` do documento, respectivamente, usando a visualização confirmada para a última.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

Para saber como consultar o campo ID do documento na visualização padrão do usuário, vá paraUsar a cláusula BY para consultar o ID do documento.

Usar a cláusula BY para consultar o ID do documento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Embora você possa definir campos destinados a serem identificadores exclusivos (por exemplo, de um veículoVIN), o verdadeiro identificador exclusivo de um documento é o campo de `id` metadados, conforme descrito em. Inserir documentos Por esse motivo, você pode usar o campo `id` para criar relações entre tabelas.

O campo `id` do documento pode ser acessado diretamente somente na visualização confirmada, mas você também pode projetá-lo na visualização padrão do usuário usando a cláusula BY. Para ver um exemplo, consulte a consulta a seguir e seus resultados.

```
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" }]
    }
}
```

Nessa consulta, `r_id` é um alias definido pelo usuário que é declarado na cláusula `FROM`, usando a palavra-chave `BY`. Esse alias `r_id` se vincula ao campo de metadados `id` de cada documento no conjunto de resultados da consulta. Você pode usar esse alias na cláusula `SELECT` e também na cláusula `WHERE` de uma consulta na visualização do usuário.

Para acessar outros atributos de metadados, no entanto, você deve consultar a visualização confirmada.

Ingressando no ID do documento

Suponha que você esteja usando o documento `id` de uma tabela como chave estrangeira em um campo definido pelo usuário de outra tabela. Você pode usar a cláusula `BY` para escrever uma consulta de junção interna para as duas tabelas nesses campos (semelhante ao Unir as visualizações confirmadas e do usuário no tópico anterior).

A consulta a seguir une duas tabelas nomeadas `DriversLicense` e `Person` em seus `PersonId` e campos `id` do documento, respectivamente, usando a cláusula `BY` para a última.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

Para saber como fazer alterações em um documento em sua tabela, vá para Atualizar e excluir documentos.

Atualizar e excluir documentos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, uma revisão de documento é uma estrutura Amazon Ion que representa uma única versão de uma sequência de documentos identificados por um ID de documento exclusivo. Cada revisão contém o conjunto de dados completo do documento, incluindo os dados do usuário e os metadados gerados pelo sistema. Cada revisão está associada a uma tabela e é identificada exclusivamente por uma combinação do ID do documento e um número de versão baseado em zero.

Quando você atualiza um documento, QLDB cria uma nova revisão com a mesma ID do documento e um número de versão incrementado. O ciclo de vida de um documento termina quando você o exclui de uma tabela. Isso significa que nenhuma revisão de documento com o mesmo ID de documento pode ser criada novamente.

Fazer revisões de documentos

Por exemplo, as instruções a seguir inserem um novo registro de veículo, atualizam a cidade de registro e, em seguida, excluem o registro. Isso resulta em três revisões de um documento.

```
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**
>
> As declarações inseridas e outras DML declarações retornam a ID de cada documento afetado. Antes de continuar, salve esse ID porque você precisa dele para a função de histórico no próximo tópico. Você também pode encontrar o ID do documento com a consulta a seguir.
>
> ```
> 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'
```

Para obter mais exemplos e informações sobre a sintaxe dessas DML declarações, consulte UPDATE e DELETE na referência do Amazon QLDB PartiQL.

Para inserir e remover elementos específicos em um documento, você pode usar `UPDATE` declarações ou outras DML declarações que comecem com a `FROM` palavra-chave. Consulte a referência de FROM(INSERT,REMOVE, ouSET) para obter mais informações e exemplos.

Depois de excluir um documento, você não poderá mais consultá-lo nas visualizações confirmada ou de usuário. Para saber como consultar o histórico de revisões deste documento usando a função de histórico integrada, vá para Consultar o histórico de revisões.

Consultar o histórico de revisões

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB armazena o histórico completo de cada documento em uma tabela. Você pode ver todas as três revisões de um documento de registro de veículo que inseriu, atualizou e excluiu anteriormente em Atualizar e excluir documentos, consultando a função de histórico integrada.

Tópicos

- Função de histórico
- Exemplo de consulta de histórico

Função de histórico

A função de histórico em QLDB é uma extensão partiQL que retorna revisões da exibição definida pelo sistema da sua tabela. Portanto, inclui seus dados e os metadados associados no mesmo esquema que a visualização confirmada.

Sintaxe

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Argumentos

table_name | '***table_id***'

O nome da tabela ou o ID da tabela. O nome da tabela é um identificador PartiQL que você pode indicar com aspas duplas ou sem aspas. O ID da tabela é um literal de string que deve estar entre aspas simples. Para saber mais sobre como usar a tabelaIDs, consulteConsultando o histórico de tabelas inativas.

`***start-time***`, `***end-time***`

(Opcional) Especifica o intervalo de tempo durante o qual todas as revisões estavam ativas. Esses parâmetros não especificam o intervalo de tempo durante o qual as revisões foram confirmadas no diário em uma transação.

Os horários de início e término são literais de timestamp de Ion que podem ser indicados com acentos graves (`` `...` ``). Para saber mais, consulte Consultando o Ion com o PartiQL na Amazon QLDB.

Esses parâmetros de tempo têm o seguinte comportamento:

- A hora de início e a hora de término são opcionais. Eles devem estar no formato de data e hora ISO8601 e no Tempo Universal Coordenado ()UTC.
- A hora de início deve ser menor ou igual à hora de término e pode ser qualquer data arbitrária no passado.
- O horário final deve ser menor ou igual à UTC data e hora atuais.
- Se você especificar uma hora de início, mas não uma hora de término, sua consulta padronizará a hora de término para a data e a hora atuais. Se você não especificar nenhum dos dois, sua consulta retornará o histórico inteiro.

'*id*'

(Opcional) O ID do documento para o qual você deseja consultar o histórico de revisões, indicado por aspas simples.

> **Tip**
>
> Como prática recomendada, qualifique uma consulta de histórico com um intervalo de datas (hora de início e hora de término) e um ID de documentos (`metadata.id`). EmQLDB, cada `SELECT` consulta é processada em uma transação e está sujeita a um limite de tempo limite de transação.
>
> As consultas de histórico não usam os índices que você cria em uma tabela. QLDBo histórico é indexado somente pela ID do documento, e você não pode criar índices de histórico adicionais no momento. As consultas de histórico que incluem uma hora de início e uma hora de término ganham o benefício da qualificação por intervalo de datas.

Exemplo de consulta de histórico

Para consultar o histórico do documento de registro do veículo, use o `id` que você salvou anteriormente em Atualizar e excluir documentos. Por exemplo, a consulta de histórico a seguir retorna todas as revisões do ID do documento `ADR2Ll1fGsU4Jr4EqTdnQF` que já estiveram ativas entre `2019-06-05T00:00:00Z` e `2019-06-05T23:59:59Z`.

> **Note**
>
> Lembre-se de que os parâmetros de hora de início e término não especificam o intervalo de tempo durante o qual as revisões foram confirmadas no diário em uma transação. Por exemplo, se uma revisão foi confirmada antes de `2019-06-05T00:00:00Z` e permaneceu ativa após esse horário de início, essa consulta de exemplo retornará essa revisão nos resultados.

Certifique-se de substituir a `id` ,hora de início e a hora de término por seus próprios valores, conforme adequado.

```
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
```

Os resultados terão uma aparência semelhante a esta:

```
{
    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"
    }
}
```

O resultado inclui atributos de metadados que fornecem detalhes sobre quando cada item foi modificado e por qual transação. A partir desses dados, você pode deduzir o seguinte:

- O documento é identificado exclusivamente por seu `id` atribuído pelo sistema: `ADR2Ll1fGsU4Jr4EqTdnQF`. UUIDIsso é representado em uma string codificada em Base62.
- Uma instrução `INSERT` cria a revisão inicial de um documento (versão `0`).
- Cada transação subsequente cria uma nova revisão com o mesmo documento `id` e um número de versão incrementado.
- O campo `txId` indica a transação que confirmou cada revisão e `txTime` mostra quando cada uma foi confirmada.
- Uma declaração `DELETE` cria uma revisão nova, mas final, de um documento. Essa revisão final tem somente metadados.

Para saber como excluir permanentemente uma revisão, vá para Redigir revisões de documentos.

Redigir revisões de documentos

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, uma `DELETE` declaração só exclui logicamente um documento criando uma nova revisão que o marca como excluído. QLDBtambém oferece suporte a uma operação de redação de dados que permite excluir permanentemente revisões de documentos inativas no histórico de uma tabela.

> **Note**
>
> No momento, todos os ledgers criados antes de 22 de julho de 2021 não são elegíveis para edição. Você pode ver a hora de criação do seu livro contábil no QLDB console da Amazon.

A operação de redação exclui somente os dados do usuário na revisão especificada e deixa a sequência do diário e os metadados do documento inalterados. Isso mantém a integridade geral dos dados do seu ledger.

Antes de começar com a redação de dadosQLDB, certifique-se de revisar Considerações e limitações de edição na referência do Amazon QLDB PartiQL.

Tópicos

- Procedimentos de redação armazenados
- Verificar se uma redação está completa
- Exemplo de redação
- Excluir e editar uma revisão ativa
- Editar um campo específico em uma revisão

Procedimentos de redação armazenados

Você pode usar o procedimento REDACT_REVISION armazenado para excluir permanentemente uma revisão individual inativa em um ledger. Esse procedimento armazenado exclui todos os dados do usuário na revisão especificada, tanto no armazenamento indexado quanto no armazenamento do diário. No entanto, deixa a sequência do diário e os metadados do documento, incluindo o ID e o hash do documento, inalterados. Essa operação é irreversível.

A revisão do documento especificada deve ser uma revisão inativa no histórico. A revisão ativa mais recente de um documento não está qualificada para redação.

Para editar várias revisões, você deve executar o procedimento armazenado uma vez para cada revisão. Você pode editar uma revisão por transação.

Sintaxe

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argumentos

`block-address`

A localização do bloco de diário da revisão do documento a ser editada. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Esse é um valor literal de Ion indicado por acentos graves. Por exemplo: .

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'*table-id*'

O ID exclusivo da tabela cuja revisão do documento você deseja redigir, indicada por aspas simples.

'*document-id*'

O ID exclusivo do documento cuja revisão será redigida, indicada por aspas simples.

Verificar se uma redação está completa

Quando você envia uma solicitação de redação executando o procedimento armazenado, QLDB processa a redação dos dados de forma assíncrona. Após a conclusão, os dados do usuário na revisão (representados pela estrutura `data`) são removidos permanentemente. Para conferir se uma solicitação de redação foi concluída, é possível usar um dos seguintes:

- Exportação de diário
- Fluxo de diário
- GetBlock APIoperação
- GetRevision APIoperação
- Função de histórico – Nota: depois que uma redação é concluída no diário, pode levar algum tempo até que as consultas do histórico mostrem o resultado da redação. Talvez você veja algumas revisões serem editadas antes de outras quando a redação assíncrona for concluída, mas as consultas ao histórico mostrarão os resultados concluídos eventualmente.

Depois que a redação de uma revisão for concluída, a estrutura `data` da revisão será substituída por um novo campo `dataHash`. O valor desse campo é o hash de Ion da estrutura `data` removida, conforme mostrado no exemplo a seguir. Como resultado, o livro mantém a integridade geral dos dados e permanece criptograficamente verificável por meio das operações de verificação existentes. API Para saber mais sobre verificação, consulte Verificação de dados na Amazon QLDB.

Exemplo de redação

Considere o documento de registro do veículo que você revisou anteriormente em Consultar o histórico de revisões. Vamos supor que você queira redigir a segunda revisão (`version:1`). O exemplo de consulta a seguir mostra essa revisão antes da redação. Nos resultados da consulta, a `data` estrutura que será editada é destacada em *`red italics`*.

```
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"
    }
}
```

Observe o `blockAddress` nos resultados da consulta porque você precisa passar esse valor para o procedimento `REDACT_REVISION` armazenado. Em seguida, encontre o ID exclusivo da tabela `VehicleRegistration`, consultando o catálogo do sistema, da seguinte forma.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Use esse ID da tabela junto com o ID do documento e o endereço do bloco para executar `REDACT_REVISION`. O ID da tabela e o ID do documento são literais de sequência de caracteres que devem estar entre aspas simples, e o endereço do bloco é um literal de Ion que está entre acentos graves. Certifique-se de substituir esses argumentos pelos seus próprios valores, conforme adequado.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> Quando você usa o QLDB console ou o QLDB shell para consultar uma ID de tabela ou ID de documento (ou qualquer valor literal de cadeia de caracteres), o valor retornado é colocado entre aspas duplas. No entanto, ao especificar os argumentos de ID da tabela e ID do documento do procedimento `REDACT_REVISION` armazenado, você deve colocar os valores entre aspas simples.
> Isso ocorre porque você escreve instruções no formato partiQL, mas QLDB retorna os resultados no formato Amazon Ion. Para obter detalhes sobre a sintaxe e a semântica do partiQL em, consulte. QLDB Consultar o Ion com o PartiQL

Uma solicitação de redação válida retorna uma estrutura de Ion que representa a revisão do documento que você está editando, da seguinte forma.

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

Quando você executa esse procedimento armazenado, QLDB processa sua solicitação de redação de forma assíncrona. Após a conclusão da redação, a `data` estrutura é removida permanentemente e substituída por uma nova *`dataHash`* campo. O valor desse campo é o hash de Ion da estrutura `data` removida.

> **Note**
>
> Esse exemplo de `dataHash` é fornecido apenas para fins informativos e não é um valor real de hash calculado.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
```

```
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Excluir e editar uma revisão ativa

As revisões ativas de documentos (ou seja, as últimas revisões não excluídas de cada documento) não são elegíveis para redação de dados. Antes de redigir uma revisão ativa, você deve primeiro atualizá-la ou excluí-la. Isso move a revisão anteriormente ativa para o histórico e a torna elegível para redação.

Se seu caso de uso exigir que todo o documento seja marcado como excluído, primeiro use uma DELETEdeclaração. Por exemplo, a instrução a seguir exclui logicamente o `VehicleRegistration` documento com um VIN de. `1HVBBAANXWH544237`

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Em seguida, edite a revisão anterior antes dessa exclusão, conforme descrito anteriormente. Se necessário, você também pode redigir individualmente quaisquer revisões anteriores.

Se seu caso de uso exigir que o documento permaneça ativo, primeiro use uma FROMinstrução UPDATEou para ocultar ou remover os campos que você deseja redigir. Esse processo é descrito na seção a seguir.

Editar um campo específico em uma revisão

QLDBnão suporta a redação de um campo específico em uma revisão do documento. Para fazer isso, você pode primeiro usar uma REMOVE instrução UPDATEFROM- REMOVE ou - para remover um campo existente de uma revisão. Por exemplo, a instrução a seguir remove o `LicensePlateNumber` campo do `VehicleRegistration` documento com um VIN de`1HVBBAANXWH544237`.

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

Em seguida, edite a revisão anterior antes dessa exclusão, conforme descrito anteriormente. Se necessário, você também pode redigir individualmente quaisquer revisões anteriores que incluam esse campo agora removido.

Para saber como otimizar suas consultas, vá para Otimizar a performance da consulta.

Otimizar a performance da consulta

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBO objetivo da Amazon é atender às necessidades de cargas de trabalho de processamento de transações on-line (OLTP) de alto desempenho. Isso significa que QLDB é otimizado para um conjunto específico de padrões de consulta, embora ofereça suporte a recursos de consulta SQL semelhantes aos de. É fundamental projetar aplicativos e seus modelos de dados para trabalhar com esses padrões de consulta. Caso contrário, à medida que suas tabelas crescerem, você encontrará problemas significativos de desempenho, incluindo latência de consultas, tempos limite de transação e conflitos de simultaneidade.

Esta seção descreve as restrições de consulta QLDB e fornece orientação para escrever consultas ideais considerando essas restrições.

Tópicos

- Tempo limite de transação
- Conflitos de concorrência
- Padrões de consulta ideais
- Padrões de consulta a serem evitados
- Monitorar o desempenho

Tempo limite de transação

EmQLDB, cada instrução partiQL (incluindo cada `SELECT` consulta) é processada em uma transação e está sujeita a um limite de tempo limite de transação. Uma transação pode ser executada por até 30 segundos antes de ser confirmada. Após esse limite, QLDB rejeita qualquer trabalho realizado na transação e descarta a sessão que executou a transação. Esse limite protege o cliente de sessões vazadas ao iniciar transações e não confirmá-las ou cancelá-las.

Conflitos de concorrência

QLDBimplementa o controle de concorrência usando o controle de concorrência otimista (). OCC Consultas abaixo do ideal também podem levar a mais conflitos. OCC Para obter mais informações sobre o OCC, consulte Modelo de QLDB concorrência da Amazon.

Padrões de consulta ideais

Como prática recomendada, você deve executar instruções com uma cláusula de predicado `WHERE` que filtre em um campo indexado ou em um ID de documento. QLDBrequer um operador de igualdade (=ou`IN`) em um campo indexado para pesquisar um documento com eficiência.

Veja a seguir exemplos de padrões de consulta ideais na visualização do usuário.

```
--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'
```

Qualquer consulta que não siga esses padrões invoca uma varredura completa da tabela. As varreduras de tabelas podem causar tempos limite de transação para consultas em tabelas grandes

ou consultas que retornam grandes conjuntos de resultados. Eles também podem levar a OCC conflitos com transações concorrentes.

Índices de alta cardinalidade

Recomendamos a indexação de campos que contenham valores de alta cardinalidade. Por exemplo, os campos `VIN` e `LicensePlateNumber` na tabela `VehicleRegistration` são campos indexados que devem ser exclusivos.

Evite indexar campos de baixa cardinalidade, como códigos de status, estados ou províncias de endereço e códigos postais. Se você indexar esse campo, suas consultas poderão produzir grandes conjuntos de resultados com maior probabilidade de resultar em tempos limite de transação ou causar conflitos não OCC intencionais.

Consultas de visualização comprometidas

As consultas que você executa na visualização confirmada seguem as mesmas diretrizes de otimização das consultas de visualização do usuário. Os índices que você cria em uma tabela também são usados para consultas na exibição confirmada.

Consultas de funções de histórico

As consultas de função de histórico não usam os índices que você cria em uma tabela. QLDBo histórico é indexado somente pela ID do documento, e você não pode criar índices de histórico adicionais no momento.

Como prática recomendada, qualifique uma consulta de histórico com um intervalo de datas (hora de início e hora de término) e um ID de documentos (`metadata.id`). As consultas de histórico que incluem uma hora de início e uma hora de término ganham o benefício da qualificação por intervalo de datas.

Consultas de junção interna

Para consultas de junção interna, use critérios de junção que incluam pelo menos um campo indexado para a tabela no lado direito da junção. Sem um índice de junção, uma consulta de junção invoca várias varreduras de tabela. Para cada documento na tabela esquerda da junção, a consulta digitaliza totalmente a tabela direita. A melhor prática é unir campos indexados para cada tabela que você está unindo, além de especificar um predicado de igualdade `WHERE` para pelo menos uma tabela.

Por exemplo, a consulta a seguir une as tabelas `VehicleRegistration` e `Vehicle` em seus respectivos campos `VIN`, ambos indexados. Essa consulta também tem um predicado de igualdade em `VehicleRegistration.VIN`.

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

Escolha índices de alta cardinalidade para os critérios de junção e os predicados de igualdade em sua consulta de junção .

Padrões de consulta a serem evitados

A seguir estão alguns exemplos de declarações abaixo do ideal que não se adaptam bem a tabelas maiores. QLDB É altamente recomendável que você não confie nesses tipos de consultas para tabelas que crescem com o tempo, pois suas consultas acabarão resultando em tempos limite de transação. Como as tabelas contêm documentos que variam em tamanho, é difícil definir limites precisos para consultas não indexadas.

```
--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)
```

Em geral, não recomendamos executar os seguintes tipos de padrões de consulta para casos de uso de produção emQLDB:

- Consultas de processamento analítico on-line (OLAP)
- Consultas exploratórias sem uma cláusula de predicado
- Consultas de relatórios
- Busca de texto

Em vez disso, recomendamos transmitir seus dados para um serviço de banco de dados com propósito específico que seja otimizado para casos de uso analítico. Por exemplo, você pode transmitir QLDB dados para o Amazon OpenSearch Service para fornecer recursos de pesquisa de texto completo em documentos. Para ver um aplicativo de amostra que demonstra esse caso de uso, consulte o GitHub repositório amazon-qldb-streaming-amazonaws-samples/ -. opensearch-service-sample-python Para obter informações sobre QLDB streams, consulteStreaming de dados de diários da Amazon QLDB.

Monitorar o desempenho

O QLDB driver fornece informações de tempo e uso de E/S consumidas no objeto resultante de uma instrução. Você pode usar essas métricas para identificar instruções PartiQL ineficientes. Para saber mais, vá para Obter estatísticas de instruções PartiQL.

Você também pode usar CloudWatch a Amazon para monitorar o desempenho do seu livro contábil para operações de dados. Monitore a métrica `CommandLatency` para um determinado `LedgerName` e `CommandType`. Para ter mais informações, consulte Monitoramento com a Amazon CloudWatch. Para saber como QLDB usa comandos para gerenciar operações de dados, consulteGerenciamento da sessão com o driver.

Obter estatísticas de instruções PartiQL

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon fornece estatísticas de execução de instruções que podem ajudá-lo a otimizar seu uso QLDB executando instruções partiQL mais eficientes. QLDBretorna essas estatísticas junto com os resultados da declaração. Eles incluem métricas que quantificam o uso de E/S consumido e o tempo de processamento do lado do servidor, que você pode usar para identificar instruções ineficientes.

Atualmente, esse recurso está disponível no editor partiQL no QLDBconsole, no QLDBshell e na versão mais recente do QLDBdriver para todas as linguagens suportadas. Você também pode visualizar estatísticas de instruções para seu histórico de consultas no console.

Tópicos

- Uso de E/S
- Informações de cronometragem

Uso de E/S

A métrica de uso de E/S descreve o número de solicitações de E/S de leitura. Se o número de solicitações de E/S de leitura for maior do que o esperado, isso indica que a instrução não está otimizada, como a falta de um índice. Recomendamos que você analise Padrões de consulta ideais no tópico anterior, Otimizar o desempenho da consulta.

> **Note**
>
> Quando você executa uma instrução `CREATE INDEX` em uma tabela não vazia, a métrica de uso de E/S inclui solicitações de leitura somente para a chamada de criação de índice síncrono.
> QLDBcria o índice para todos os documentos existentes na tabela de forma assíncrona. Essas solicitações de leitura assíncrona não estão incluídas na métrica de uso de E/S dos resultados da sua declaração. As solicitações de leitura assíncrona são cobradas separadamente e adicionadas ao total de E/S de leitura após a conclusão da criação do índice.

Usar o console de QLDB

Para obter o uso de E/S de leitura de uma instrução usando o QLDB console, siga as seguintes etapas:

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, selecione Editor PartiQL.
3. Escolha um ledger na lista suspensa.
4. Na janela do editor de consulta, insira a instrução de sua escolha e, em seguida, selecione Executar. Veja um exemplo de consulta a seguir:

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Para executar a instrução, você também pode usar o atalho de teclado Ctrl+Enter para Windows ou Cmd+Return para macOS. Para obter mais atalhos de teclado, consulte Atalhos de teclado do editor PartiQL.
5. Abaixo da janela do editor de consultas, os resultados da consulta incluem E/S de leitura, que é o número de solicitações de leitura feitas pela instrução.

Você também pode visualizar as I/Os de leitura do seu histórico de consultas seguindo as seguintes etapas:

1. No painel de navegação, escolha Consultas recentes no editor partiQL.
2. A coluna E/Ss de leitura exibe o número de solicitações de leitura feitas por cada instrução.

Usando o QLDB driver

Para obter o uso de E/S de uma instrução usando o QLDB driver, chame a `getConsumedIOs` operação do cursor de fluxo ou do cursor em buffer do resultado.

Os exemplos de código a seguir mostram como obter E/S lidas do cursor de fluxo do resultado de uma instrução.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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))
```

Os exemplos de código a seguir mostram como obter E/S lidas do cursor de fluxo do resultado de uma instrução. Isso retorna o total de I/Os de leitura e solicitações `ExecuteStatement` e `FetchPage`.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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**
>
> O cursor do stream tem estado porque pagina o conjunto de resultados. Portanto, as operações `getConsumedIOs` e `getTimingInformation` retornam as métricas acumuladas a partir do momento em que você as chama.
> O cursor armazenado em buffer armazena em buffer o conjunto de resultados na memória e retorna o total de métricas acumuladas.

Informações de cronometragem

A métrica de informações de tempo descreve o tempo de processamento do lado do servidor em milissegundos. O tempo de processamento do lado do servidor é definido como a quantidade de tempo QLDB gasto no processamento de uma declaração. Isso não inclui o tempo gasto em chamadas ou pausas na rede. Essa métrica separa o tempo de processamento no lado do QLDB serviço do tempo de processamento no lado do cliente.

Usar o console de QLDB

Para obter as informações de tempo de uma declaração usando o QLDB console, siga as seguintes etapas:

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, selecione Editor PartiQL.
3. Escolha um ledger na lista suspensa.
4. Na janela do editor de consulta, insira a instrução de sua escolha e, em seguida, selecione Executar. Veja um exemplo de consulta a seguir:

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Para executar a instrução, você também pode usar o atalho de teclado Ctrl+Enter para Windows ou Cmd+Return para macOS. Para obter mais atalhos de teclado, consulte Atalhos de teclado do editor PartiQL.
5. Abaixo da janela do editor de consultas, os resultados da consulta incluem latência do lado do servidor, que é a quantidade de tempo entre o QLDB recebimento da solicitação de instrução e o envio da resposta. Esse é um subconjunto da duração total da consulta.

Você também pode visualizar as E/Ss de leitura do seu histórico de consultas seguindo as seguintes etapas:

1. No painel de navegação, escolha Consultas recentes no editor partiQL.
2. A coluna Tempo de execução (ms) exibe essas informações de tempo para cada instrução.

Usando o QLDB driver

Para obter as informações de tempo de uma instrução usando o QLDB driver, chame a `getTimingInformation` operação do cursor de fluxo ou do cursor em buffer do resultado.

Os exemplos de código a seguir mostram como obter E/S lidas do cursor de fluxo do resultado de uma instrução.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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))
```

Os exemplos de código a seguir mostram como obter o tempo de processamento do cursor de fluxo do resultado de uma instrução. Isso retorna o total de processamento de solicitações `ExecuteStatement` e `FetchPage`.

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**
>
> Para converter em código síncrono, remova as palavras-chave `await` e `async` e altere o tipo `IAsyncResult` para `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**
>
> O cursor do stream tem estado porque pagina o conjunto de resultados. Portanto, as operações `getConsumedIOs` e `getTimingInformation` retornam as métricas acumuladas a partir do momento em que você as chama.
> O cursor armazenado em buffer armazena em buffer o conjunto de resultados na memória e retorna o total de métricas acumuladas.

Para saber como consultar o catálogo do sistema, vá para Consultar o catálogo do sistema.

Consultar o catálogo do sistema

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Cada tabela que você cria em um QLDB livro contábil da Amazon tem uma ID exclusiva atribuída pelo sistema. Você pode encontrar o ID de uma tabela, sua lista de índices e outros metadados consultando a tabela `information_schema.user_tables` do catálogo do sistema.

Todos os sistemas atribuídos IDs são identificadores universalmente exclusivos (UUID), cada um representado em uma string codificada em Base62. Para ter mais informações, consulte Único IDs na Amazon QLDB.

O exemplo a seguir mostra os resultados de uma consulta que retorna atributos de metadados da tabela `VehicleRegistration`.

```
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"
}
```

Campos de metadados de tabelas

- `tableId`: o ID exclusivo da tabela.
- `name`: o nome da tabela.
- `indexes`: a lista de índices na tabela.
  - `indexId`: o ID exclusivo do índice.
  - `expr`: o caminho do documento indexado. Esse campo é uma string no formato: `[fieldName]`.
  - `status`: o status atual do índice (`BUILDING`, `FINALIZING`, `ONLINE`, `FAILED` ou`DELETING`). QLDBnão usa o índice nas consultas até que o status seja`ONLINE`.
  - `message`: a mensagem de erro que descreve o motivo pelo qual o índice tem um status `FAILED`. Esse campo só é incluído para índices com falha.
- `status`: o status atual da tabela (`ACTIVE` ou `INACTIVE`). Uma tabela se torna `INACTIVE` quando você a `DROP`.

Para saber como gerenciar tabelas usando as instruções `DROP TABLE` e `UNDROP TABLE`, vá para Gerenciar tabelas.

Gerenciar tabelas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve como gerenciar tabelas usando as `UNDROP TABLE` instruções `DROP TABLE` e na AmazonQLDB. Também descreve como marcar tabelas enquanto você as cria. As cotas para o número de tabelas ativas e o total de tabelas que você pode criar são definidas em Cotas e limites na Amazon QLDB.

Tópicos

- Marcar tabelas na criação
- Eliminar tabelas
- Consultando o histórico de tabelas inativas
- Reativando tabelas

Marcar tabelas na criação

> **Note**
>
> Atualmente, a marcação de tabelas na criação é suportada apenas para ledgers no modo de permissões `STANDARD`.

Você já pode marcar os recursos da tabela. Para gerenciar tags para tabelas existentes, use as API operações AWS Management Console `TagResource` ou `ListTagsForResource` e. `UntagResource` Para ter mais informações, consulte Marcação de recursos da Amazon QLDB.

Você também pode definir tags de tabela ao criar a tabela usando o QLDB console ou especificando-as em uma instrução `CREATE TABLE` partiQL. O exemplo a seguir cria uma tabela chamada `Vehicle` com a tag `environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Ao marcar os recursos no momento da criação, você elimina a necessidade de executar scripts personalizados de marcação após a criação do recurso. Depois que uma tabela é marcada, você pode controlar o acesso à tabela com base nessas tags. Por exemplo, você pode conceder acesso total somente a tabelas que tenham uma tag específica. Para obter um exemplo JSON de política, consulteAcesso total a todas as ações com base nas tags da tabela.

Eliminar tabelas

Para eliminar uma tabela, use uma instrução básica DROP TABLE. Quando você coloca uma mesaQLDB, você está apenas desativando-a.

Por exemplo, a instrução a seguir desativa a tabela `VehicleRegistration`.

```
DROP TABLE VehicleRegistration
```

Uma instrução `DROP TABLE` retorna o ID atribuído pelo sistema da nova tabela. O status de `VehicleRegistration` agora deve ser `INACTIVE` na tabela do catálogo do sistema information_schema.user_tables.

```
SELECT status FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Consultando o histórico de tabelas inativas

Além do nome da tabela, você também pode consultar o QLDB Função de histórico com um ID da tabela como primeiro argumento de entrada. Você deve usar o ID da tabela para consultar o histórico de uma tabela inativa. Depois que uma tabela é desativada, você não pode mais consultar seu histórico com o nome da tabela.

Primeiro, encontre o ID da tabela consultando a tabela do catálogo do sistema. Por exemplo, a consulta a seguir retorna o `tableId` da tabela `VehicleRegistration`.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Em seguida, você pode usar esse ID para executar a mesma consulta de histórico de Consultar o histórico de revisões. Veja a seguir um exemplo que consulta o histórico do ID do documento a

`ADR2Ll1fGsU4Jr4EqTdnQF` partir do ID da tabela. `5PLf9SXwndd63lPaSIa0O6`. O ID da tabela é um literal de string que deve estar entre aspas simples.

```
--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'
```

Reativando tabelas

Depois de desativar uma tabela emQLDB, você pode usar a UNDROP TABLE instrução para reativá-la.

Primeiro, encontre o ID da tabela em `information_schema.user_tables`. Por exemplo, a consulta a seguir retorna o `tableId` da tabela `VehicleRegistration`. O status deveria ser `INACTIVE`.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Em seguida, use esse ID para reativar a tabela. Veja a seguir um exemplo que revela o ID da tabla `5PLf9SXwndd63lPaSIa0O6`. Nesse caso, o ID da tabela é um identificador exclusivo que você coloca entre aspas duplas.

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

O status de agora `VehicleRegistration` deveria ser`ACTIVE`.

Para saber como criar, descrever e eliminar índices, vá para Gerenciamento de Índices.

Gerenciamento de Índices

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve como criar, descrever e eliminar índices na AmazonQLDB. A cota para o número de índices por tabela que você pode criar é definida em Cotas e limites na Amazon QLDB.

Tópicos

- Criar índices
- Descrever índices
- Reduzir índices
- Erros comuns

Criar índices

Conforme também descrito emCriar tabelas e índices, você pode usar a CREATEINDEXinstrução para criar um índice em uma tabela para um campo de nível superior especificado, da seguinte maneira. O nome da tabela e o nome do campo indexado diferenciam maiúsculas de minúsculas.

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

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

Cada índice que você cria em um ledger do Amazon QLDB tem um ID exclusivo atribuído pelo sistema. Para encontrar esse ID de índice, consulte a seção Descrever índices a seguir.

> **Important**
>
> QLDBrequer um índice para pesquisar um documento com eficiência. Sem um índice, QLDB precisa fazer uma digitalização completa da tabela ao ler documentos. Isso pode causar problemas de desempenho em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
>
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade (= ou `IN`) em um campo indexado ou em um ID de documento. Para ter mais informações, consulte Otimizar a performance da consulta.

Observe as seguintes restrições ao criar índices:

- Um índice só pode ser criado em um único campo de nível superior. Não há suporte para índices compostos, aninhados, exclusivos e baseados em funções.
- Você pode criar um índice em qualquer tipo de dados Ion, incluindo `list` e `struct`. No entanto, você só pode fazer a pesquisa indexada pela igualdade de todo o valor de Ion, independentemente do tipo de Ion. Por exemplo, ao usar um tipo `list` como índice, você não pode fazer uma pesquisa indexada por um item dentro da lista.
- O desempenho da consulta é aprimorado somente quando você usa um predicado de igualdade; por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`.

  QLDBnão respeita as desigualdades nos predicados de consulta. Como resultado, as verificações filtradas por intervalo não são implementadas.
- Os nomes dos campos indexados diferenciam maiúsculas e minúsculas e podem ter no máximo 128 caracteres.
- A criação do índice em QLDB é assíncrona. O tempo necessário para concluir a criação de um índice em uma tabela não vazia varia dependendo do tamanho da tabela. Para ter mais informações, consulte Gerenciamento de Índices.

Descrever índices

A criação do índice em QLDB é assíncrona. O tempo necessário para concluir a criação de um índice em uma tabela não vazia varia dependendo do tamanho da tabela. Para verificar o status de uma criação de índice, você pode consultar a tabela do catálogo do sistema information_schema.user_tables.

Por exemplo, a instrução a seguir consulta o catálogo do sistema para todos os índices na tabela `VehicleRegistration`.

```
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"
}
```

campos de índice

- `indexId`: o ID exclusivo do índice.
- `expr`: o caminho do documento indexado. Esse campo é uma string no formato: `[fieldName]`.
- `status`: o status atual do índice. O status de um índice pode ser um dos valores a seguir:
  - `BUILDING`: está construindo ativamente o índice para a tabela.
  - `FINALIZING`: concluiu a construção do índice e está começando a ativá-lo para uso.
  - `ONLINE`: está ativo e pronto para uso em consultas. QLDBnão usa o índice em consultas até que o status esteja on-line.
  - `FAILED`: não é possível criar o índice devido a um erro irrecuperável. Os índices nesse estado ainda contam para sua cota de índices por tabela. Para ter mais informações, consulte Erros comuns.
  - `DELETING`: está excluindo ativamente o índice depois que um usuário o excluiu.
- `message`: a mensagem de erro que descreve o motivo pelo qual o índice tem um status `FAILED`. Esse campo só é incluído para índices com falha.

Usar o console

Você também pode usar o AWS Management Console para verificar o status de um índice.

Como verificar o status de um índice (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista Ledgers, escolha o nome do ledger cujos índices você deseja gerenciar.
4. Na página de detalhes do ledger, na guia Tabelas, escolha o nome da tabela cujo índice você deseja verificar.
5. Na página de detalhes da tabela, localize o cartão Campos indexados. A coluna Status do índice exibe o status atual de cada índice na tabela.

Reduzir índices

Use a instrução DROP INDEX para eliminar um índice. Quando você descarta um índice, ele é excluído permanentemente da tabela.

Primeiro, encontre o ID da tabela em `information_schema.user_tables`. Por exemplo, a consulta a seguir retorna o `indexId` do campo `LicensePlateNumber` indexado na tabela `VehicleRegistration`.

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

Em seguida, use esse ID para eliminar o índice. Veja a seguir um exemplo que revela o ID do índice `4tPW3fUhaVhDinRgKRLhGU`. O ID do índice é um identificador exclusivo que você coloca entre aspas duplas.

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> A cláusula `WITH (purge = true)` é obrigatória para todas as instruções `DROP INDEX` e atualmente `true` é o único valor suportado.
> A palavra-chave `purge` diferencia maiúsculas e minúsculas e deve ser toda em minúsculas.

Usar o console

Você também pode usar o AWS Management Console para remover um índice.

Para eliminar um índice (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista Ledgers, escolha o nome do ledger cujos índices você deseja gerenciar.
4. Na página de detalhes do ledger, na guia Tabelas, escolha o nome da tabela cujo índice você deseja verificar.

5. Na página de detalhes da tabela, localize o cartão Campos indexados. Selecione o índice que você deseja eliminar e escolha Eliminar índice.

Erros comuns

Esta seção descreve erros comuns que você pode encontrar ao criar índices e sugere possíveis soluções.

> **Note**
>
> Os índices com status de `FAILED` ainda contam para sua cota de índices por tabela. Um índice com falha também impede que você modifique ou exclua qualquer documento que tenha causado a falha na criação do índice na tabela.
> Você deve eliminar explicitamente o índice para removê-lo da cota.

O documento contém vários valores para o campo indexado: ***`fieldName`***.

QLDBnão consegue criar um índice para o nome do campo especificado porque a tabela contém um documento com vários valores para o mesmo campo (ou seja, nomes de campo duplicados).

Você deve primeiro eliminar o índice com falha. Em seguida, certifique-se de que todos os documentos na tabela tenham somente um valor para cada nome de campo antes de tentar criar o índice novamente. Você também pode criar um índice para outro campo que não tenha duplicatas.

QLDBtambém retorna esse erro se você tentar inserir um documento que contém vários valores para um campo que já está indexado na tabela.

Limite de índices excedido: Tabela ***`tableName`*** já tem ***`n`*** índices e não é possível criar mais.

QLDBimpõe um limite de cinco índices por tabela, incluindo índices com falha. Você deve eliminar um índice existente antes de criar um novo.

Nenhum índice definido com identificador: ***`indexId`***.

Você tentou eliminar um índice que não existe para a combinação especificada de tabela e ID de índice. Para saber como verificar os índices existentes, consulte Descrever índices.

Único IDs na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve as propriedades e as diretrizes de uso dos identificadores exclusivos atribuídos pelo sistema na Amazon. QLDB Ele também fornece alguns exemplos de itens QLDB exclusivosIDs.

Tópicos

- Propriedades
- Uso
- Exemplos

Propriedades

Todos os sistemas atribuídos IDs em QLDB são identificadores universalmente exclusivos (). UUID Cada geometria tem as seguintes propriedades:

- Número de 128 bits UUID
- Representado em texto codificado em Base62
- Sequência alfanumérica de comprimento fixo de 22 caracteres (por exemplo: `3Qv67yjXEwB9SjmvkuG6Cp`)

Uso

Ao usar o QLDB unique IDs em seu aplicativo, observe as seguintes diretrizes:

Faça

- Trate o ID como uma string.

Não

- Tente decodificar a string.
- Atribua um significado semântico à string (como derivar um componente de tempo).
- Classifique as sequências em uma ordem semântica.

Exemplos

Os atributos a seguir são alguns exemplos de exclusividade IDs emQLDB:

- ID do documento
- ID do índice
- ID da vertente
- ID da tabela
- ID da transação.

Modelo de QLDB concorrência da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBO objetivo da Amazon é atender às necessidades de cargas de trabalho de processamento de transações on-line (OLTP) de alto desempenho. QLDBoferece suporte a recursos SQL de consulta semelhantes aos de consultas e fornece ACID transações completas. Além disso, QLDB os itens de dados são documentos, oferecendo flexibilidade de esquema e modelagem de dados intuitiva. Com um diário no centro, você pode usá-lo QLDB para acessar o histórico completo e verificável de todas as alterações em seus dados e transmitir transações coerentes para outros serviços de dados, conforme necessário.

Tópicos

- Controle de simultaneidade otimista
- Usar índices para evitar varreduras completas da tabela
- Conflitos de inserção OCC
- Tornar as transações idempotentes
- Conflitos de redação OCC
- Gerenciar sessões simultâneas

Controle de simultaneidade otimista

EmQLDB, o controle de concorrência é implementado usando o controle de concorrência otimista (). OCC OCCopera com base no princípio de que várias transações podem ser concluídas frequentemente sem interferir umas nas outras.

UsandoOCC, as transações QLDB não adquirem bloqueios nos recursos do banco de dados e operam com isolamento serializável total. QLDBexecuta transações simultâneas em série, de forma que produza o mesmo efeito como se essas transações tivessem sido iniciadas em série.

Antes da confirmação, cada transação executa uma verificação de validação para garantir que nenhuma outra transação confirmada tenha modificado os dados que está acessando. Se essa verificação revelar modificações conflitantes ou se o estado dos dados mudar, a transação confirmada será rejeitada. No entanto, a transação pode ser reiniciada.

Quando uma transação grava emQLDB, as verificações de validação do OCC modelo são implementadas por QLDB si mesmas. Se uma transação não puder ser gravada no diário devido a uma falha na fase de verificação doOCC, QLDB retorna uma `OccConflictException` para a camada de aplicação. O software do aplicativo é responsável por garantir que a transação seja reiniciada. O aplicativo deve interromper a transação rejeitada e, em seguida, repetir toda a transação desde o início.

Para saber como o QLDB driver lida e repete OCC conflitos e outras exceções transitórias, consulte. Entendendo a política de repetição com o motorista na Amazon QLDB

Usar índices para evitar varreduras completas da tabela

EmQLDB, cada instrução partiQL (incluindo cada `SELECT` consulta) é processada em uma transação e está sujeita a um limite de tempo limite de transação.

Como prática recomendada, você deve executar instruções com uma cláusula de predicado `WHERE` que filtre em um campo indexado ou em um ID de documento. QLDBrequer um operador de igualdade em um campo indexado para pesquisar um documento com eficiência; por exemplo, `WHERE indexedField = 123` ou`WHERE indexedField IN (456, 789)`.

Sem essa pesquisa indexada, QLDB precisa fazer uma varredura completa da tabela ao ler documentos. Isso pode causar latência de consultas e tempos limite de transação, além de aumentar as chances de um OCC conflito com transações concorrentes.

Por exemplo, considere uma tabela chamada `Vehicle` que tenha um índice somente no campo `VIN`. Contém os seguintes documentos.

VIN	Make	Modelo	Cor
`"1N4AL11D 75C109151"`	`"Audi"`	`"A5"`	`"Silver"`
`"KM8SRDHF 6EU074761"`	`"Tesla"`	`"Model S"`	`"Blue"`

VIN	Make	Modelo	Cor
"3HGGK5G5 3FM761765"	"Ducati"	"Monster 1200"	"Yellow"
"1HVBBAAN XWH544237"	"Ford"	"F 150"	"Black"
"1C4RJFAG 0FC625797"	"Mercedes"	"CLK 350"	"White"

Dois usuários concomitantes chamados Alice e Bob estão trabalhando com a mesma tabela em um ledger. Eles querem atualizar dois documentos diferentes, da seguinte forma.

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'
```

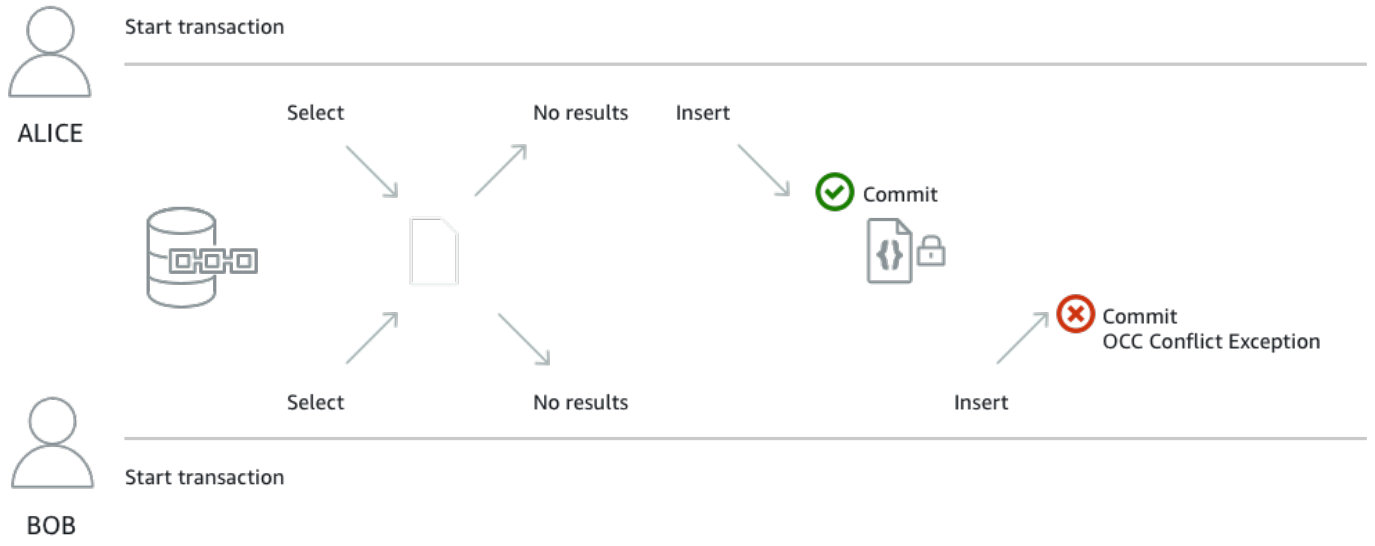
Suponha que Alice e Bob iniciem suas transações ao mesmo tempo. A instrução `UPDATE` de Alice faz uma pesquisa indexada no campo `VIN`, então ela só precisa ler aquele documento. Alice termina e confirma sua transação com êxito primeiro.

A instrução de Bob filtra em campos não indexados, então ela faz uma varredura da tabela e encontra um `OccConflictException`. Isso ocorre porque a transação confirmada de Alice modificou os dados que a instrução de Bob está acessando, o que inclui todos os documentos da tabela, não apenas o documento que Bob está atualizando.

Conflitos de inserção OCC

OCCos conflitos podem incluir documentos que foram inseridos recentemente, não apenas documentos que existiam anteriormente. Considere o diagrama a seguir, no qual dois usuários

concomitantes (Alice e Bob) estão trabalhando com a mesma tabela em um ledger. Ambos desejam inserir um novo documento somente sob a condição de que ainda não exista um valor de predicado.



Neste exemplo, tanto Alice quanto Bob executam as instruções `SELECT` e `INSERT` a seguir, em uma única transação. Seu aplicativo executa a instrução `INSERT` somente se a instrução `SELECT` não retornar resultados.

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

Suponha que Alice e Bob iniciem suas transações ao mesmo tempo. Ambas as consultas `SELECT` não retornam nenhum documento existente com um `VIN` de `ABCDE12345EXAMPLE`. Então, seus aplicativos prosseguem com a instrução `INSERT`.

Alice termina e confirma sua transação com êxito primeiro. Então, Bob tenta confirmar sua transação, mas a QLDB rejeita e lança uma. `OccConflictException` Isso ocorre porque a transação

confirmada de Alice modificou o conjunto de resultados da `SELECT` consulta de Bob e OCC detecta esse conflito antes de confirmar a transação de Bob.

A consulta `SELECT` é necessária para que esse exemplo de transação seja idempotente. Bob pode então repetir toda a transação desde o início. Mas sua próxima consulta `SELECT` retornará o documento que Alice inseriu, então o aplicativo de Bob não executará o `INSERT`.

Tornar as transações idempotentes

A transação de inserção na seção anterior também é um exemplo de transação idempotente. Em outras palavras, executar a mesma transação várias vezes produz resultados idênticos. Se Bob executar o `INSERT` sem primeiro verificar se um determinado `VIN` já existe, a tabela pode acabar com documentos com valores `VIN` duplicados.

Considere outros cenários de repetição, além dos OCC conflitos. Por exemplo, é possível que uma transação seja confirmada QLDB com sucesso no lado do servidor, mas o cliente expire enquanto espera por uma resposta. Recomendamos que você torne as transações de gravação idempotentes para evitar efeitos colaterais inesperados no caso de simultaneidade ou tentativas repetidas.

Conflitos de redação OCC

QLDBevita redações simultâneas de revisões no mesmo bloco de diário. Considere um exemplo em que dois usuários concomitantes (Alice e Bob) desejam redigir duas revisões de documentos diferentes que são confirmadas no mesmo bloco em um ledger. Primeiro, Alice solicita a redação de uma revisão executando o procedimento armazenado `REDACT_REVISION`, da seguinte forma.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Então, enquanto a solicitação de Alice ainda está sendo processada, Bob solicita a redação de outra revisão, da seguinte forma.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDBrejeita a solicitação de Bob `OccConflictException` mesmo que eles estejam tentando redigir duas revisões de documentos diferentes. Isso ocorre porque a revisão de Bob está localizada

no mesmo bloco da revisão que Alice está editando. Depois que a solicitação de Alice terminar o processamento, Bob poderá repetir sua solicitação de redação.

Da mesma forma, se duas transações simultâneas tentarem redigir a mesma revisão, somente uma solicitação poderá ser processada. A outra solicitação falha com uma exceção de OCC conflito até que a redação seja concluída. Posteriormente, qualquer solicitação para redigir a mesma revisão resultará em um erro que indica que a revisão já foi editada.

Gerenciar sessões simultâneas

Se você tiver experiência no uso de um sistema de gerenciamento de banco de dados relacional (RDBMS), talvez esteja familiarizado com os limites de conexão simultânea. QLDBnão tem o mesmo conceito de RDBMS conexão tradicional porque as transações são executadas com mensagens de HTTP solicitação e resposta.

EmQLDB, o conceito análogo é uma sessão ativa. Uma sessão é conceitualmente semelhante a um login de usuário — ela gerencia as informações sobre suas solicitações de transação de dados em um ledger. Uma sessão ativa é aquela que está executando ativamente uma transação. Também pode ser uma sessão que concluiu recentemente uma transação em que o serviço prevê que iniciará outra transação imediatamente. QLDBsuporta uma transação em execução ativa por sessão.

O limite de sessões ativas simultâneas por ledger é definido em Cotas e limites na Amazon QLDB. Depois que esse limite for atingido, qualquer sessão que tente iniciar uma transação resultará em um erro (`LimitExceededException`).

Para obter informações sobre o ciclo de vida de uma sessão e como o QLDB driver lida com as sessões ao executar transações de dados, consulte. Gerenciamento da sessão com o driver Para obter as melhores práticas para configurar um pool de sessões em seu aplicativo usando o QLDB driver, consulte as recomendações Configurando o objeto QldbDriver de QLDB drivers da Amazon.

Verificação de dados na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Com a AmazonQLDB, você pode confiar que o histórico de alterações nos dados do seu aplicativo é preciso. QLDBusa um registro transacional imutável, conhecido como diário, para armazenamento de dados. O diário rastreia todas as alterações em seus dados confirmados e mantém um histórico de alterações completo e verificável ao longo do tempo.

QLDBusa a função de hash SHA -256 com um modelo baseado em árvore Merkle para gerar uma representação criptográfica do seu diário, conhecida como resumo. O resumo funciona como uma assinatura exclusiva de todo o histórico de alterações dos seus dados em um determinado momento. Você usa o resumo para verificar a integridade das revisões do documento em relação a essa assinatura.

Tópicos

- Em que tipo de dados você pode verificarQLDB?
- O que significa integridade de dados?
- Como a verificação funciona?
- Exemplo de verificação
- Como a redação de dados afeta a verificação?
- Introdução à verificação
- Etapa 1: Solicitar um resumo em QLDB
- Etapa 2: Verificar seus dados em QLDB
- Resultados da verificação
- Tutorial: Verificando dados usando um AWS SDK
- Erros comuns de verificação

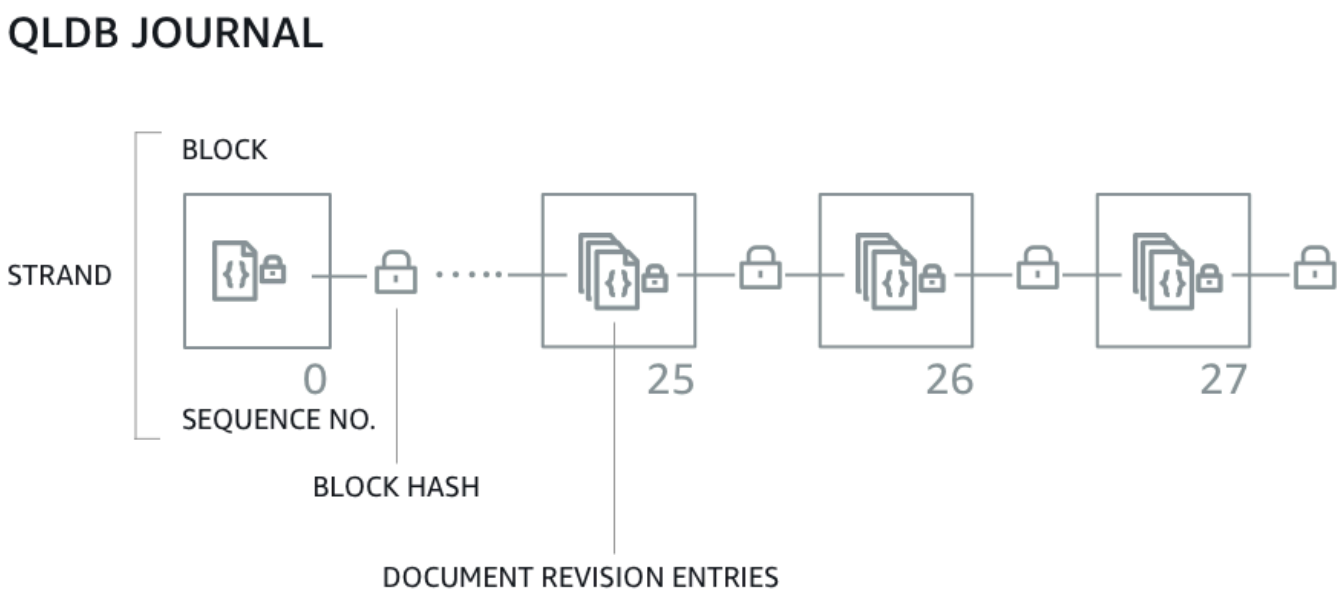
Em que tipo de dados você pode verificarQLDB?

EmQLDB, cada livro contábil tem exatamente um diário. Um diário pode ter várias vertentes, que são partições do diário.

> **Note**
>
> QLDBatualmente suporta periódicos com apenas uma única vertente.

Bloco é um objeto que é comprometido com a cadeia do diário durante uma transação. Esse bloco contém objetos de entrada, que representam as revisões do documento que resultaram da transação. Você pode verificar uma revisão individual ou um bloco inteiro do diárioQLDB.

O diagrama a seguir ilustra essa estrutura do diário.



O diagrama mostra que as transações são confirmadas no diário como blocos que contêm entradas de revisão do documento. Também mostra que cada bloco está encadeado em hash aos blocos subsequentes, e tem um número de sequência para especificar seu endereço na cadeia.

Para obter mais informações sobre o conteúdo de dados em um bloco, consulte Conteúdo do periódico na Amazon QLDB.

O que significa integridade de dados?

A integridade dos dados QLDB significa que o diário do seu livro é, de fato, imutável. Em outras palavras, seus dados (especificamente, cada revisão de documento) estão em um estado em que o seguinte é verdadeiro:

1. Ele existe no mesmo local em seu diário em que foi escrito pela primeira vez.
2. Não foi alterado de forma alguma desde que foi escrito.

Como a verificação funciona?

Para entender como a verificação funciona na AmazonQLDB, você pode dividir o conceito em quatro componentes básicos.

- Hashing
- Resumo
- Árvore Merkle
- Prova

Hashing

QLDBusa a função hash criptográfica SHA -256 para criar valores de hash de 256 bits. Um hash atua como uma assinatura exclusiva de tamanho fixo de qualquer quantidade arbitrária de dados de entrada. Se você alterar qualquer parte da entrada, mesmo um único caractere ou bit, o hash de saída mudará completamente.

O diagrama a seguir mostra que a função de hash SHA -256 cria valores de hash completamente exclusivos para dois QLDB documentos que diferem em apenas um único dígito.

```
vehicle = {
            'VIN' :       "KM8SRDHF6EU074761",
            'Type':       "Truck"
            'Model':      "F150"
            'Specs': {
                  'EngSize' : 3.3
                  'CurbWeight': 4,878
                  'HP': 327
            }
}
```

SHA-256 → a4e31e36910d99bd19b7f875f0a04597dc0ff52c2f164a16a9288aed9e710fdd

```
vehicle = {
            'VIN' :       "KM8SRDHF6EU074761",
            'Type':       "Truck"
            'Model':      "F150"
            'Specs': {
                  'EngSize' : 3.3
                  'CurbWeight': 4,879
                  'HP': 327
            }
}
```

SHA-256 → 19318457408920af2d2cbeacd90c7afe0fbd7f6ff316972c8f656c8bbc402dd1

A função hash SHA -256 é unidirecional, o que significa que não é matematicamente viável calcular a entrada quando recebe uma saída. O diagrama a seguir mostra que não é possível calcular o QLDB documento de entrada quando se recebe um valor de hash de saída.

```
vehicle = {
            'VIN' :       "KM8SRDHF6EU074761",
            'Type':       "Truck"
            'Model':      "F150"
            'Specs': {
                  'EngSize' : 3.3
                  'CurbWeight': 4,878
                  'HP': 327
            }
}
```

SHA-256 → a4e31e36910d99bd19b7f875f0a04597dc0ff52c2f164a16a9288aed9e710fdd ✓

SHA-256 ← 19318457408920af2d2cbeacd90c7afe0fbd7f6ff316972c8f656c8bbc402dd1

As seguintes entradas de dados são inseridas QLDB para fins de verificação:

- Revisões do documento
- Instruções em PartiQL
- Entradas de revisão
- Blocos de diário

Resumo

Um resumo é uma representação criptográfica de todo o diário do seu livro em um determinado momento. Um diário é somente para anexar, e os blocos de diário são sequenciados e encadeados em hash de forma semelhante aos blockchains.

Você pode solicitar um resumo para um ledger a qualquer momento. QLDBgera o resumo e o retorna para você como um arquivo de saída seguro. Em seguida, você usa esse resumo para verificar a integridade das revisões de documentos que foram confirmadas em um momento anterior. Se você recalcular os hashes começando com uma revisão e terminando com o resumo, você prova que seus dados não foram alterados no meio.

Árvore Merkle

À medida que o tamanho do seu ledger aumenta, torna-se cada vez mais ineficiente recalcular toda a cadeia de hash do diário para verificação. QLDBusa um modelo de árvore Merkle para lidar com essa ineficiência.

Uma árvore Merkle é uma estrutura de dados em árvore na qual cada nó da folha representa um hash de um bloco de dados. Cada nó não foliar é um hash de seus nós secundários. Comumente usada em blockchains, uma árvore Merkle ajuda você a verificar com eficiência grandes conjuntos de dados com um mecanismo à prova de auditoria. Para obter mais informações sobre árvores Merkle, consulte a página da Wikipedia sobre árvores Merkle. Para saber mais sobre as provas de auditoria da Merkle e um exemplo de caso de uso, consulte Como funcionam as provas de log no site Certificate Transparency.

A QLDB implementação da árvore Merkle é construída a partir da cadeia de hash completa de um diário. Nesse modelo, os nós da folha são o conjunto de todos os hashes de revisão de documentos individuais. O nó raiz representa o resumo de todo o diário em um determinado momento.

Usando uma prova de auditoria Merkle, você pode verificar uma revisão verificando apenas um pequeno subconjunto do histórico de revisões do seu ledger. Você faz isso atravessando a árvore de um determinado nó da folha (revisão) até sua raiz (resumo). Ao longo desse caminho de travessia, você faz o hash recursivo de pares de nós irmãos para calcular o hash principal até terminar com o resumo. Essa travessia tem uma complexidade temporal de nós `log(n)` na árvore.

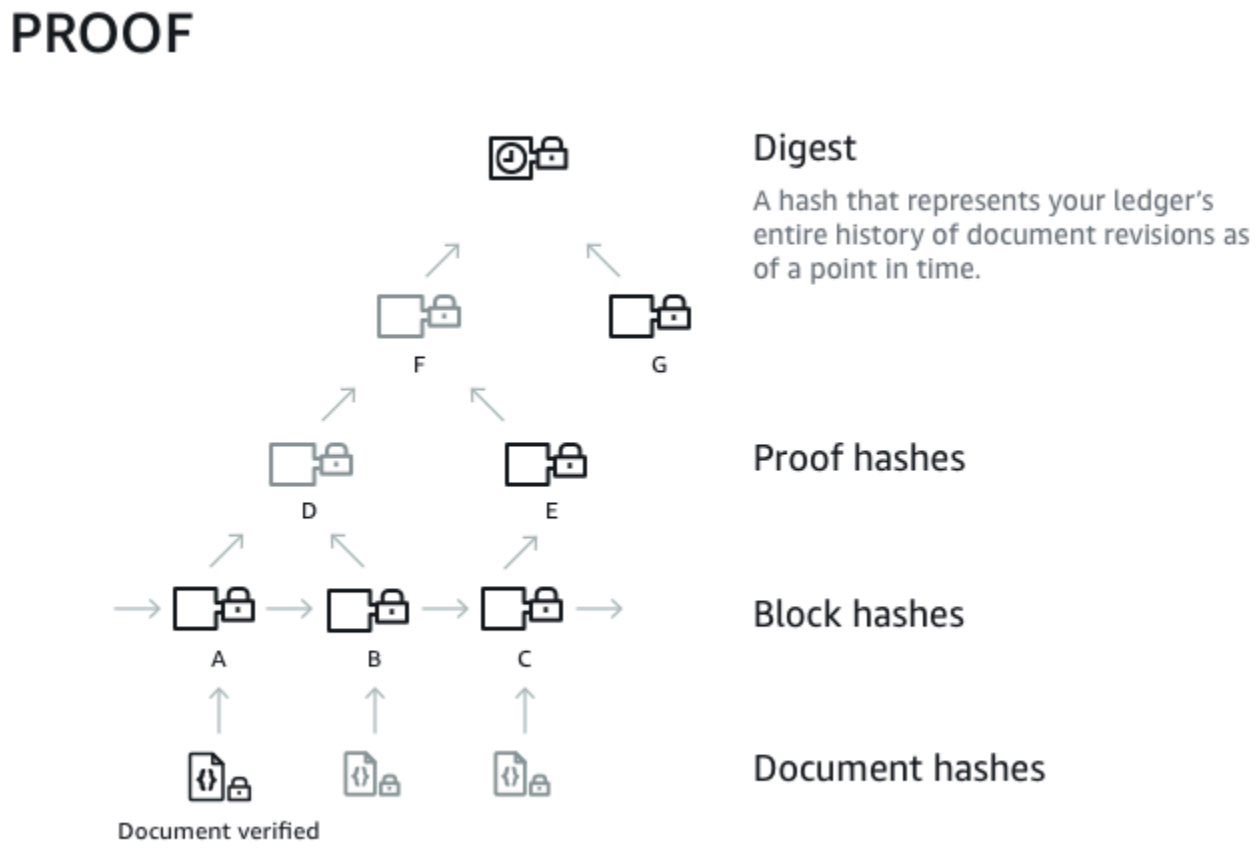
Prova

Uma prova é a lista ordenada de hashes de nós que QLDB retorna para um determinado resumo e revisão do documento. Ele consiste nos hashes exigidos por um modelo de árvore Merkle para encadear o hash do nó foliar fornecido (uma revisão) ao hash raiz (o resumo).

Alterar qualquer dado confirmado entre uma revisão e um resumo quebra a cadeia de hash do seu diário e impossibilita a geração de uma prova.

Exemplo de verificação

O diagrama a seguir ilustra o modelo de árvore de QLDB hash da Amazon. Ele mostra um conjunto de hashes de blocos que se acumulam até o nó raiz superior, que representa o resumo de uma sequência do diário. Em um ledger com um diário de fita única, esse nó raiz também é o resumo de todo o ledger.



Suponha que o nó A seja o bloco que contém a revisão do documento cujo hash você deseja verificar. Os nós a seguir representam a lista ordenada de hashes QLDB fornecida em sua prova: B, E, G. Esses hashes são necessários para recalcular o resumo do hash A.

Para recalcular o resumo, faça o seguinte:

1. Comece com o hash A e concatene-o com o hash B. Em seguida, faça o hash do resultado para calcular D.
2. Use D e E para calcular F.
3. Use F e G para calcular o resumo.

A verificação será bem-sucedida se o resumo recalculado corresponder ao valor esperado. Com um hash de revisão e um resumo, não é possível fazer engenharia reversa dos hashes em uma prova. Portanto, este exercício prova que sua revisão foi realmente escrita neste local do diário em relação ao resumo.

Como a redação de dados afeta a verificação?

Na AmazonQLDB, uma `DELETE` declaração só exclui logicamente um documento criando uma nova revisão que o marca como excluído. QLDBtambém oferece suporte a uma operação de redação de dados que permite excluir permanentemente revisões de documentos inativas no histórico de uma tabela.

A operação de redação exclui somente os dados do usuário na revisão especificada e deixa a sequência do diário e os metadados do documento inalterados. Depois que uma revisão é redigida, os dados do usuário na revisão (representados pela estrutura `data`) são substituídos por um novo campo `dataHash`. O valor desse campo é o hash de Amazon Ion da estrutura `data` removida. Para obter mais informações e um exemplo de uma operação de redação, consulte Redigir revisões de documentos.

Como resultado, o livro mantém a integridade geral dos dados e permanece criptograficamente verificável por meio das operações de verificação existentes. API Você ainda pode usar essas API operações conforme o esperado para solicitar um resumo (GetDigest), solicitar uma prova (GetBlockou GetRevision) e, em seguida, executar seu algoritmo de verificação usando os objetos retornados.

Como recalcular um hash de revisão

Se você planeja verificar uma revisão de documento individual recalculando seu hash, você deve verificar condicionalmente se a revisão foi editada. Se a revisão foi redigida, você pode usar o valor de hash fornecido no campo `dataHash`. Se não tiver sido editado, você pode recalcular o hash usando o campo `data`.

Ao fazer essa verificação condicional, você pode identificar as revisões editadas e tomar as medidas apropriadas. Por exemplo, você pode registrar eventos de manipulação de dados para fins de monitoramento.

Introdução à verificação

Antes de verificar os dados, você deve solicitar um resumo do seu ledger e salvá-lo para mais tarde. Qualquer revisão de documento que seja confirmada antes do último bloco coberto pelo resumo é elegível para verificação em relação a esse resumo.

Em seguida, você solicita um comprovante da Amazon QLDB para uma revisão qualificada que deseja verificar. Usando essa prova, você chama um cliente API para recalcular o resumo, começando com o hash da revisão. Desde que o resumo salvo anteriormente seja conhecido e confiável externamenteQLDB, a integridade do seu documento será comprovada se o hash do resumo recalculado corresponder ao hash do resumo salvo.

> **Important**
>
> - O que você está provando especificamente é que a revisão do documento não foi alterada entre o momento em que você salvou esse resumo e o momento em que executou a verificação. Você pode solicitar e salvar um resumo assim que uma revisão que você deseja verificar posteriormente for enviada para o diário.
> - Como prática recomendada, recomendamos que você solicite resumos regularmente e guarde-os fora do ledger. Determine a frequência com que você solicita resumos com base na frequência com que você confirma as revisões em seu ledger.
>
> Para um detalhado AWS postagem no blog que discute o valor da verificação criptográfica no contexto de um caso de uso realista, consulte Verificação criptográfica no mundo real com a Amazon. QLDB

Para obter step-by-step guias sobre como solicitar um resumo do seu livro contábil e depois verificar seus dados, consulte o seguinte:

- Etapa 1: Solicitar um resumo em QLDB
- Etapa 2: Verificar seus dados em QLDB

Etapa 1: Solicitar um resumo em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon fornece um API formulário para solicitar um resumo que cubra a ponta atual do diário em seu livro contábil. A dica do diário se refere ao último bloco confirmado no momento em que QLDB recebe sua solicitação. Você pode usar o AWS Management Console, um AWS SDK, ou o AWS Command Line Interface (AWS CLI) para obter um resumo.

Tópicos

- AWS Management Console
- QLDB API

AWS Management Console

Siga estas etapas para solicitar um resumo usando o QLDB console.

Para solicitar um resumo (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista de ledgers, selecione o nome do ledger para o qual você deseja solicitar um resumo.
4. Escolha Obter resumo. A caixa de diálogo Obter resumo exibe os seguintes detalhes do resumo:
   - Digest — O valor de hash SHA -256 do resumo que você solicitou.

- Endereço da dica de resumo: a última localização do bloco no diário coberto pelo resumo que você solicitou. Um endereço tem os dois campos a seguir:
  - `strandId`: o ID exclusivo da cadeia do diário que contém o bloco.
  - `sequenceNo`: o número do índice que especifica a localização do bloco dentro da cadeia.
- ledger: o nome do ledger para o qual você solicitou um resumo.
- Data: a data e hora em que você solicitou o resumo.

5. Analise as informações do resumo. Em seguida, escolha Salvar. Você pode manter o nome do arquivo padrão ou inserir um novo nome.

> **Note**
>
> Você pode notar que seus valores de hash de resumo e endereço de dica mudam mesmo quando você não modifica nenhum dado em seu ledger. Isso ocorre porque o console recupera o catálogo do sistema do ledger toda vez que você executa uma consulta no editor partiQL. Essa é uma transação de leitura que é confirmada no diário e faz com que o endereço do bloco mais recente seja alterado.

Essa etapa salva um arquivo de texto simples com conteúdo no formato Amazon Ion. O arquivo tem uma extensão de nome de arquivo `.ion.txt` e contém todas as informações resumidas listadas na caixa de diálogo anterior. Este é um exemplo do conteúdo de um arquivo de resumo: A ordem dos campos pode variar dependendo do seu navegador.

```
{
  "digest": "42zaJOfV8iGutVGNaIuzQWhD5Xb/5B9lScHnvxPXm9E=",
  "digestTipAddress": "{strandId:\"BlFTjlSXze9BIh1KOszcE3\",sequenceNo:73}",
  "ledger": "my-ledger",
  "date": "2019-04-17T16:57:26.749Z"
}
```

6. Salve esse arquivo onde você possa acessá-lo futuramente. Posteriormente, você pode usar esse arquivo para verificar a revisão de um documento.

> **Important**
>
> A revisão do documento que você verifica posteriormente deve ser coberta pelo resumo que você salvou. Ou seja, o número de sequência do endereço do documento deve ser menor ou igual ao número de sequência do endereço da dica de resumo.

QLDB API

Você também pode solicitar um resumo do seu livro contábil usando a Amazon QLDB API com um AWS SDKou o AWS CLI. O QLDB API fornece a seguinte operação para uso por programas aplicativos:

- GetDigest— Retorna o resumo de um livro contábil no último bloco confirmado no diário. A resposta inclui um valor de hash de 256 bits e um endereço do bloco.

Para obter informações sobre como solicitar um resumo usando o AWS CLI, veja o comando get-digest no AWS CLI Referência de comando.

Aplicação de exemplo

Para exemplos de código Java, consulte o GitHub repositório amazon-qldb-dmv-sampleaws-samples/ -java. Para obter instruções sobre como baixar e instalar esse aplicativo de amostra, consulte Instalando o aplicativo de amostra Amazon QLDB Java. Antes de solicitar um resumo, siga as etapas 1 a 3 do Tutorial de Java para criar um livro de amostras e carregá-lo com dados de amostra.

O código do tutorial na aula GetDigestfornece um exemplo de solicitação de um resumo do livro de `vehicle-registration` amostra.

Para verificar a revisão de um documento usando o resumo que você salvou, vá para Etapa 2: Verificar seus dados em QLDB.

Etapa 2: Verificar seus dados em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon fornece uma API para solicitar um comprovante para um ID de documento especificado e seu bloco associado. Você também deve fornecer o endereço da dica de um resumo que você salvou anteriormente, conforme descrito em Etapa 1: Solicitar um resumo em QLDB. Você pode usar o AWS Management Console, um AWS SDK, ou o AWS CLI para obter uma prova.

Em seguida, você pode usar a prova retornada por QLDB para verificar a revisão do documento em relação ao resumo salvo, usando o lado do clienteAPI. Isso permite que você controle o algoritmo usado para verificar seus dados.

Tópicos

- AWS Management Console
- QLDB API

AWS Management Console

Esta seção descreve as etapas para verificar uma revisão de documento em relação a um resumo salvo anteriormente usando o QLDB console da Amazon.

Antes de começar, certifique-se de seguir as etapas em Etapa 1: Solicitar um resumo em QLDB. A verificação requer um resumo salvo anteriormente que cubra a revisão que você deseja verificar.

Para verificar a revisão de um documento (console)

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. Primeiro, consulte seu ledger para o `id` e `blockAddress` da revisão que você deseja verificar. Esses campos estão incluídos nos metadados do documento, que você pode consultar na visualização confirmada.

O documento `id` é uma sequência de caracteres de identificação exclusiva atribuída pelo sistema. `blockAddress` é uma estrutura Ion que especifica a localização do bloco onde a revisão foi confirmada.

No painel de navegação, selecione Editor PartiQL.

3. Escolha o nome do livro no qual você deseja verificar uma revisão.
4. Na janela do editor de consultas insira uma instrução `SELECT` na sintaxe abaixo e escolha Executar.

```
SELECT metadata.id, blockAddress FROM _ql_committed_table_name
WHERE criteria
```

Por exemplo, a consulta a seguir retorna um documento da tabela `VehicleRegistration` no ledger de amostra criado em Começando a usar o QLDB console da Amazon.

```
SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = 'KM8SRDHF6EU074761'
```

5. Copie e salve os valores `id` e `blockAddress` que sua consulta devolve. Certifique-se de omitir as aspas duplas do campo `id`. No Amazon Ion, os tipos de dados de string são delimitados com aspas duplas. Por exemplo, você deve copiar somente o texto alfanumérico no trecho a seguir.

   `"LtMNJYNjSwzBLgf7sLifrG"`

6. Agora que você selecionou uma revisão do documento, pode iniciar o processo de verificação.

   No painel de navegação, escolha Verificação.

7. No formulário Verificar documento, em Especificar o documento que deseja verificar, insira os seguintes parâmetros de entrada:
   - ledger: o ledger no qual você deseja verificar uma revisão.
   - Endereço do bloco: o valor `blockAddress` devolvido por sua consulta na etapa 4.
   - ID do documento: o valor `id` devolvido por sua consulta na etapa 4.
8. Em Especificar o resumo a ser usado para verificação, selecione o resumo que você salvou anteriormente escolhendo Escolher resumo. Se o arquivo for válido, isso preencherá automaticamente todos os campos de resumo no console. Ou você pode copiar e colar manualmente os seguintes valores diretamente do seu arquivo de resumo:

- Resumo: o valor `digest` do seu arquivo de resumo.
- Endereço dica de resumo: o valor `digestTipAddress` do seu arquivo de resumo.

9. Revise os parâmetros de entrada do documento e do resumo e escolha Verificar.

   O console automatiza duas etapas para você:

   a. Solicite um formulário de prova QLDB para o documento especificado.
   b. Use o comprovante retornado por QLDB para ligar para um clienteAPI, que verifica a revisão do documento em relação ao resumo fornecido. Para examinar esse algoritmo de verificação, consulte a seção a seguir QLDB API para baixar o exemplo de código.

   O console exibe os resultados da sua solicitação no cartão Resultados da verificação. Para obter mais informações, consulte Resultados da verificação.

QLDB API

Você também pode verificar a revisão de um documento usando a Amazon QLDB API com um AWS SDKou o AWS CLI. O QLDB API fornece as seguintes operações para uso por programas aplicativos:

- `GetDigest`: devolve o resumo de um ledger no último bloco confirmado no diário. A resposta inclui um valor de hash de 256 bits e um endereço do bloco.
- `GetBlock`: devolve um objeto de bloco em um endereço especificado em um diário. Também devolve uma prova do bloco especificado para verificação, se `DigestTipAddress` for fornecida.
- `GetRevision`: devolve um objeto de dados de revisão para um ID de documento e endereço de bloco especificados. Também devolve uma prova da revisão especificada para verificação, se `DigestTipAddress` for fornecida.

Para obter descrições completas dessas API operações, consulte QLDBAPIReferência da Amazon o.

Para obter informações sobre a verificação de dados usando o AWS CLI, veja o AWS CLI Referência de comando.

Aplicação de exemplo

Para exemplos de código Java, consulte o GitHub repositório amazon-qldb-dmv-sampleaws-samples/ -java. Para obter instruções sobre como baixar e instalar esse aplicativo de amostra, consulte Instalando o aplicativo de amostra Amazon QLDB Java. Antes de fazer uma verificação, siga as etapas de 1 a 3 no Tutorial de Java para criar um livro de amostras e carregá-lo com dados de amostra.

O código do tutorial na aula GetRevisionfornece um exemplo de solicitação de uma prova para uma revisão de documento e, em seguida, verificação dessa revisão. Essa classe executa as seguintes etapas:

1. Solicita um novo resumo do ledger `vehicle-registration` de amostra.
2. Solicita uma prova para uma amostra de revisão de documento da tabela `VehicleRegistration` no ledger `vehicle-registration`.
3. Verifica a revisão da amostra usando o resumo e a prova devolvidos.

Resultados da verificação

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve os resultados retornados por uma solicitação de verificação de QLDB dados da Amazon no AWS Management Console. Para obter etapas detalhadas sobre como enviar uma solicitação de verificação, consulteEtapa 2: Verificar seus dados em QLDB.

Na página de verificação do QLDB console, os resultados da sua solicitação são exibidos no cartão de resultados da verificação. A guia Prova mostra o conteúdo da prova retornada QLDB para a revisão e resumo do documento especificado. Isso inclui os seguintes detalhes:

- Hash de revisão — O valor SHA -256 que representa de forma exclusiva a revisão do documento que você está verificando.

- Hashes de prova — A lista ordenada de hashes fornecida por QLDB isso é usada para recalcular o resumo especificado. O console começa com o Hash de revisão e o combina sequencialmente com cada hash de prova até terminar com um resumo recalculado.

  A lista é reduzida por padrão, então você pode expandi-la para revelar os valores de hash. Opcionalmente, você mesmo pode testar os cálculos de hash seguindo as etapas descritas em Como usar uma prova para recalcular seu resumo.
- Resumo calculado: o hash que resultou da série de cálculos de hash que foram feitos no hash de revisão. Se esse valor corresponder ao resumo salvo anteriormente, a verificação será bem-sucedida.

A guia Bloquear mostra o conteúdo do bloco que contém a revisão que você está verificando. Isso inclui os seguintes detalhes:

- ID da transação: o ID exclusivo da transação que confirmou esse bloqueio.
- Hora da transação: a data e hora em que esse bloco foi comprometido com a cadeia.
- Hash do bloco — O valor SHA -256 que representa exclusivamente esse bloco e todo o seu conteúdo.
- Endereço do bloco: o local no diário do seu ledger em que esse bloqueio foi confirmado. Um endereço tem os dois campos a seguir:
  - ID da cadeia: a identificação exclusiva da cadeia do diário que contém esse bloco.
  - Número de sequência: o número do índice que especifica a localização desse bloco dentro da cadeia.
- Declarações: as instruções partiQL que foram executadas para confirmar entradas nesse bloco.

> **Note**
>
> Se você executar instruções parametrizadas programaticamente, elas serão registradas em seus blocos de diário com parâmetros de associação em vez de dados literais. Por exemplo, você pode ver a seguinte instrução em um bloco de diário, em que o ponto de interrogação (?) é um marcador variável para o conteúdo do documento.
>
> ```
> INSERT INTO Vehicle ?
> ```

- Entradas do documento: as revisões do documento que foram confirmadas neste bloco.

Se sua solicitação não conseguiu verificar a revisão do documento, consulte Erros comuns de verificação para obter informações sobre possíveis causas.

Como usar uma prova para recalcular seu resumo

Depois de QLDB devolver um comprovante para sua solicitação de verificação de documentos, você mesmo pode tentar fazer os cálculos de hash. Esta seção descreve as etapas de alto nível para recalcular seu resumo usando a prova fornecida.

Primeiro, emparelhe seu Hash de revisão com o primeiro hash na lista de hashes de prova. Então, faça o seguinte:

1. Classifique os dois hashes. Compare os hashes por seus valores de bytes assinados em ordem little-endian.
2. Concatene os dois hashes em ordem ordenada.
3. Faça o hash do par concatenado com um SHA gerador de hash -256.
4. Combine seu novo hash com o próximo hash da prova e repita as etapas 1 a 3. Depois de processar o hash de última prova, seu novo hash é seu resumo recalculado.

Se o resumo recalculado corresponder ao resumo salvo anteriormente, seu documento será verificado com sucesso.

Para ver um step-by-step tutorial com exemplos de código que demonstram essas etapas de verificação, vá paraTutorial: Verificando dados usando um AWS SDK.

Tutorial: Verificando dados usando um AWS SDK

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Neste tutorial, você verifica um hash de revisão de documento e um hash de bloco de diário em um QLDB livro contábil da Amazon usando o por meio de um QLDB API AWS SDK. Você também usa o QLDB driver para consultar a revisão do documento.

Considere um exemplo em que você tem uma revisão de documento que contém dados de um veículo com um número de identificação do veículo (VIN) de`KM8SRDHF6EU074761`. A revisão do documento está em uma tabela `VehicleRegistration` que está em um ledger chamado `vehicle-registration`. Suponha que você queira verificar a integridade da revisão do documento desse veículo e do bloco de diário que contém a revisão.

> **Note**
>
> Para um detalhado AWS postagem no blog que discute o valor da verificação criptográfica no contexto de um caso de uso realista, consulte Verificação criptográfica no mundo real com a Amazon. QLDB

Tópicos

- Pré-requisitos
- Etapa 1: Solicite um resumo
- Etapa 2: Consultar a revisão do documento
- Etapa 3: Solicitar uma prova da revisão
- Etapa 4: Recalcular o resumo da revisão
- Etapa 5: Solicitar uma prova para o bloco de diário
- Etapa 6: Recalcular o resumo do bloco
- Execute o exemplo de código completo

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Configure o QLDB driver para um idioma de sua escolha preenchendo os respectivos pré-requisitos abaixo. Começando com o QLDB driver da Amazon Isso inclui se inscrever em AWS, concedendo acesso programático para desenvolvimento e configurando seu ambiente de desenvolvimento.
2. Siga as etapas 1 a 2 em Começando a usar o QLDB console da Amazon para criar um ledger chamado `vehicle-registration` e carregá-lo com dados de amostra predefinidos.

Em seguida, revise as etapas a seguir para saber como a verificação funciona e, em seguida, execute o exemplo de código completo do início ao fim.

Etapa 1: Solicite um resumo

Antes de verificar os dados, você deve primeiro solicitar um resumo do seu ledger `vehicle-registration` para uso posterior.

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')
```

Etapa 2: Consultar a revisão do documento

Use o QLDB driver para consultar os endereços de blocos, hashes e documentos IDs associados a. VIN `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)
```

Etapa 3: Solicitar uma prova da revisão

Repita os resultados da consulta e use cada endereço de bloco e ID do documento junto com o nome do ledger para enviar uma solicitação `GetRevision`. Para obter uma prova da revisão, você também deve fornecer o endereço da dica do resumo salvo anteriormente. Essa API operação retorna um objeto que inclui a revisão do documento e a prova da revisão.

Para obter mais informações sobre a estrutura e o conteúdo da revisão, consulte Consultar metadados do documento.

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)
```

Em seguida, recupere a prova da revisão solicitada.

O QLDB API retorna a prova como uma representação em sequência da lista ordenada de hashes de nós. Para converter essa string em uma lista da representação binária dos hashes dos nós, você pode usar um leitor de íons da biblioteca Amazon Ion. Para obter mais informações sobre o uso da biblioteca Ion, consulte o Cookbook Amazon Ion.

Java

Neste exemplo, você usa `IonReader` para fazer a conversão binária.

```
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

Neste exemplo, você usa `IonLoader` para carregar a prova em um datagrama de íons.

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

Neste exemplo, você usa um leitor de íons para converter a prova em binária e para percorrer a lista de hashes de nós da prova.

```
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

Neste exemplo, você usa a função `load` para fazer a conversão binária.

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

Neste exemplo, você usa a função `loads` para fazer a conversão binária.

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Etapa 4: Recalcular o resumo da revisão

Use a lista de hashes da prova para recalcular o resumo, começando com o hash da revisão. Desde que o resumo salvo anteriormente seja conhecido e confiável fora da empresaQLDB, a integridade da revisão do documento é comprovada se o hash do resumo recalculado corresponder ao hash do resumo salvo.

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))
```

Etapa 5: Solicitar uma prova para o bloco de diário

Em seguida, você verifica o bloco de diário que contém a revisão do documento.

Use o endereço do bloco e o endereço da dica do resumo que você salvou na Etapa 1 para enviar uma solicitação `GetBlock`. Semelhante à solicitação `GetRevision` na Etapa 2, você deve fornecer novamente o endereço da dica do resumo salvo para obter uma prova do bloco. Essa API operação retorna um objeto que inclui o bloco e a prova do bloco.

Para obter informações sobre a estrutura do bloco de diário e seu conteúdo, consulte Conteúdo do periódico na 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)
```

Em seguida, recupere o hash do bloco e a prova do resultado.

Java

Neste exemplo, você usa `IonLoader` para carregar o objeto de bloco em um contêiner `IonDatagram`.

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

Você também usa `IonLoader` para carregar a prova em um `IonDatagram`.

```
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

Neste exemplo, você usa `IonLoader` para carregar o bloco e a prova em um datagrama de íons para cada um.

```
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

Neste exemplo, você usa um leitor de íons para converter a prova em binária e para percorrer a lista de hashes de nós da prova.

```
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

Neste exemplo, você usa a função `load` para converter o bloco e a prova em binário.

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

Neste exemplo, você usa a função `loads` para converter o bloco e a prova em binário.

```
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)
```

Etapa 6: Recalcular o resumo do bloco

Use a lista de hashes da prova para recalcular o resumo, começando com o hash do bloco. Desde que o resumo salvo anteriormente seja conhecido e confiável fora da empresaQLDB, a integridade do bloco é comprovada se o hash do resumo recalculado corresponder ao hash do resumo salvo.

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))
```

Os exemplos de código anteriores usam a função `dot` a seguir ao recalcular o resumo. Essa função recebe uma entrada de dois hashes, os classifica, concatena e retorna o hash da matriz concatenada.

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
```

Execute o exemplo de código completo

Execute o exemplo de código completo da seguinte forma para realizar todas as etapas anteriores do início ao fim.

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))
```

Erros comuns de verificação

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve os erros de tempo de execução que são lançados pela Amazon QLDB para solicitações de verificação.

Veja a seguir uma lista de exceções comuns retornadas pelo serviço. Cada exceção inclui a mensagem de erro específica, seguida pelas API operações que podem gerá-la, uma breve descrição e sugestões de possíveis soluções.

IllegalArgumentException

Mensagem: O valor de Ion fornecido não é válido e não pode ser analisado.

APIoperações: `GetDigest, GetBlock, GetRevision`

Certifique-se de fornecer um valor válido do Amazon Ion antes de tentar novamente sua solicitação.

IllegalArgumentException

Mensagem: O endereço de bloco fornecido não é válido.

APIoperações: `GetDigest, GetBlock, GetRevision`

Certifique-se de fornecer um valor válido de endereço de bloco antes de tentar novamente sua solicitação. Um endereço de bloco é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Mensagem: O número de sequência do endereço da dica de resumo fornecido está além do último registro confirmado da cadeia.

APIoperações: `GetDigest, GetBlock, GetRevision`

O endereço da dica de resumo que você fornecer deve ter um número de sequência menor ou igual ao número de sequência do último registro confirmado da vertente do diário. Antes de repetir sua solicitação, certifique-se de fornecer um endereço de sugestão com um número de sequência válido.

IllegalArgumentException

Mensagem: O ID de cadeia do endereço de bloco fornecido não é válido.

APIoperações: `GetDigest, GetBlock, GetRevision`

O endereço de bloco que você fornece deve ter um ID de cadeia que corresponda ao ID da cadeia do diário. Certifique-se de fornecer um valor válido de endereço de bloco antes de tentar novamente com um ID de cadeia.

IllegalArgumentException

Mensagem: O número de sequência do endereço de bloco fornecido está além do último registro confirmado da cadeia.

APIoperações: `GetBlock, GetRevision`

O endereço de bloco que você fornece deve ter um número de sequência menor ou igual ao número de sequência do último registro confirmado da cadeia. Antes de repetir sua solicitação, certifique-se de fornecer um endereço de bloco com um número de sequência válido.

IllegalArgumentException

Mensagem: O ID de cadeia do endereço de bloco fornecido deve corresponder ao ID de cadeia do endereço da dica de resumo fornecido.

APIoperações: `GetBlock, GetRevision`

Você só pode verificar uma revisão ou bloco de documento se ele existir na mesma linha do diário que o resumo fornecido.

IllegalArgumentException

Mensagem: O número de sequência do endereço de bloco fornecido não deve ser maior que o número de sequência do endereço de dica de resumo fornecido.

APIoperações: `GetBlock, GetRevision`

Você só pode verificar uma revisão ou bloco de documento se estiver coberto pelo mesmo resumo que você forneceu. Isso significa que foi entregue ao diário antes do endereço da dica de resumo.

IllegalArgumentException

Mensagem: O ID do documento fornecido não foi encontrado no bloco no endereço de bloco especificado.

APIoperação: `GetRevision`

O ID do documento que você fornece deve existir no endereço de bloco fornecido. Antes de repetir sua solicitação, verifique se esses dois parâmetros são consistentes.

Exportação de dados de periódicos da Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB usa um registro transacional imutável, conhecido como diário, para armazenamento de dados. O diário rastreia todas as alterações em seus dados confirmados e mantém um histórico de alterações completo e verificável ao longo do tempo.

Você pode acessar o conteúdo do diário em seu livro para vários fins, incluindo análise, auditoria, retenção de dados, verificação e exportação para outros sistemas. Os tópicos a seguir descrevem como exportar blocos de diário do seu livro contábil para um bucket do Amazon Simple Storage Service (Amazon S3) em seu Conta da AWS. Um trabalho de exportação de diário grava seus dados no Amazon S3 como objetos na representação de texto ou binária do formato Amazon Ion ou no formato de texto JSONLines.

No formato JSON Linhas, cada bloco em um objeto de dados exportado é um JSON objeto válido delimitado por uma nova linha. Você pode usar esse formato para integrar diretamente JSON as exportações com ferramentas de análise, como Amazon Athena e AWS Glue porque esses serviços podem analisar automaticamente os delimitados por novas linhasJSON. Para obter mais informações sobre o formato, consulte JSONLinhas.

Para obter informações sobre o Amazon S3, consulte o Guia do usuário do Amazon Simple Storage Service.

> **Note**
>
> Se você especificar JSON como formato de saída do seu trabalho de exportação, QLDB converterá negativamente os dados do diário Ion JSON em seus objetos de dados exportados. Para obter mais informações, consulte Conversão descendente para JSON.

Tópicos

- Solicitando a exportação de um diário em QLDB
- Saída de exportação do diário em QLDB
- Permissões de exportação de diário em QLDB
- Erros comuns na exportação de diários

Solicitando a exportação de um diário em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon fornece uma API opção para solicitar a exportação de seus blocos de diário para um intervalo de data e hora especificado e um destino específico do bucket do Amazon S3. Um trabalho de exportação de diário pode gravar os objetos de dados no texto ou na representação binária do formato Amazon Ion ou no formato de texto JSONLines. Você pode usar o AWS Management Console, um AWS SDK, ou o AWS Command Line Interface (AWS CLI) para criar um trabalho de exportação.

Tópicos

- AWS Management Console
- QLDB API
- Expiração do trabalho de exportação

AWS Management Console

Siga estas etapas para enviar uma solicitação de exportação de diário QLDB usando o QLDB console.

Para solicitar uma exportação (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.

2. No painel de navegação, selecione Exportar.
3. Escolha Criar exportação.
4. Na página Criar tarefa de exportação, insira as seguintes configurações de exportação:

   - ledger: o ledger cujos blocos de diário você deseja exportar.
   - Data e hora de início — A data e hora de início inclusiva no Tempo Universal Coordenado (UTC) do intervalo de blocos de diário a serem exportados. Esse carimbo de data/hora deve ser anterior à data e hora de término. Se você fornecer um registro de data e hora de início anterior ao do livro contábil`CreationDateTime`, o QLDB padrão será o do livro contábil. `CreationDateTime`
   - Data e hora de término — O carimbo de data/hora de término exclusivo (UTC) da variedade de blocos de diário a serem exportados. Essa data e hora não podem estar no futuro.
   - Destino dos blocos de diário: o bucket do Amazon S3 e o nome do prefixo no qual seu trabalho de exportação grava os objetos de dados. Use o seguinte formato do Amazon S3URI.

     ```
     s3://amzn-s3-demo-bucket/prefix/
     ```

     É necessário especificar um nome de bucket do S3 e um nome de prefixo opcional para os objetos de saída. Veja um exemplo a seguir.

     ```
     s3://amzn-s3-demo-bucket/journalExport/
     ```

     O nome e o prefixo do bucket devem estar em conformidade com as regras e convenções de nomenclatura do Amazon S3. Para obter informações sobre como nomear buckets, consulte Restrições e limitações de buckets no Guia do usuário do Amazon S3. Para ter mais informações sobre prefixos de nomes de chaves, consulte Chave de objeto e metadados.

     > **Note**
     >
     > Exportações entre regiões não são compatíveis. O bucket do Amazon S3 especificado deve estar no mesmo Região da AWS como seu livro contábil.

   - Criptografia S3: as configurações de criptografia usadas pelo seu trabalho de exportação para gravar dados em um bucket do Amazon S3. Para obter mais informações sobre o uso da criptografia no lado do servidor no Amazon S3, consulte Proteger dados usando criptografia no lado do servidor, no Guia do desenvolvedor do Amazon S3.

- Criptografia padrão do bucket: use as configurações de criptografia padrão do bucket do Amazon S3 especificado.
- AES-256 — Use criptografia do lado do servidor com chaves gerenciadas do Amazon S3 (-S3). SSE
- AWS- KMS — Use criptografia do lado do servidor com AWS KMS chaves gerenciadas (SSE-KMS).

  Se você escolher esse tipo junto com a opção Escolha um diferente AWS KMS keyopção, você também deve especificar uma KMS chave de criptografia simétrica no seguinte formato Amazon Resource Name (ARN).

  ```
  arn:aws:kms:aws-region:account-id:key/key-id
  ```

- Acesso ao serviço — A IAM função que concede permissões de QLDB gravação em seu bucket do Amazon S3. Se aplicável, a IAM função também deve conceder QLDB permissões para usar sua KMS chave.

  Para passar uma função QLDB ao solicitar a exportação de um diário, você deve ter permissões para realizar a `iam:PassRole` ação no recurso da IAM função.

  - Crie e use um novo perfil de serviço: deixe o console criar uma nova função para você com as permissões necessárias para o bucket do Amazon S3 especificado.
  - Use uma função de serviço existente — Para saber como criar essa função manualmente emIAM, consultePermissões de exportação.
- Formato de saída: o formato de saída dos dados exportados do diário
  - Texto de íons: (padrão) representação de texto do Amazon Ion
  - Binário de íons: representação binária do Amazon Ion
  - JSON— Formato de texto delimitado por nova linha JSON

    Se você escolherJSON, QLDB converte negativamente os dados do diário Ion JSON em seus objetos de dados exportados. Para obter mais informações, consulte Conversão descendente para JSON.

5. Quando estiver satisfeito com as configurações, escolha Criar trabalho de exportação.

   O tempo necessário para a conclusão do trabalho de exportação varia dependendo do tamanho dos dados. Se o envio da solicitação for bem-sucedido, o console retornará à página principal de Exportação e listará seus trabalhos de exportação com o status atual.

6. É possível ver seus objetos de exportação no console do Amazon S3.

   Abra o console do Amazon S3 em. https://console.aws.amazon.com/s3/

   Para saber mais sobre o formato desses objetos de saída, consulte Saída de exportação do diário em QLDB.

> **Note**
>
> Os trabalhos de exportação expiram sete dias após serem concluídos. Para obter mais informações, consulte Expiração do trabalho de exportação.

QLDB API

Você também pode solicitar a exportação de um periódico usando a Amazon QLDB API com um AWS SDKou o AWS CLI. O QLDB API fornece as seguintes operações para uso por programas aplicativos:

- `ExportJournalToS3`: exporta o conteúdo do diário dentro de um intervalo de data e hora de um determinado ledger para um bucket específico do Amazon S3. Um trabalho de exportação pode gravar os dados como objetos na representação de texto ou binária do formato Amazon Ion ou no formato de texto JSON Lines.
- `DescribeJournalS3Export`: devolve informações detalhadas sobre um trabalho de exportação de diário. A saída inclui seu status atual, hora de criação e os parâmetros da sua solicitação de exportação original.
- `ListJournalS3Exports`— Retorna uma lista de descrições de trabalhos de exportação de periódicos para todos os livros contábeis associados ao atual Conta da AWS e região. A saída de cada descrição do trabalho de exportação inclui os mesmos detalhes retornados por `DescribeJournalS3Export`.
- `ListJournalS3ExportsForLedger`: devolve uma lista de descrições de trabalhos de exportação de diários para um determinado ledger. A saída de cada descrição do trabalho de exportação inclui os mesmos detalhes retornados por `DescribeJournalS3Export`.

Para obter descrições completas dessas API operações, consulte QLDBAPIReferência da Amazon o.

Para obter informações sobre a exportação de dados do diário usando o AWS CLI, veja o AWS CLI Referência de comando.

Aplicativos de exemplo (Java)

Para exemplos de código Java de operações básicas de exportação, consulte o GitHub repositório amazon-qldb-dmv-sampleaws-samples/ -java. Para obter instruções sobre como baixar e instalar esse aplicativo de amostra, consulte Instalando o aplicativo de amostra Amazon QLDB Java. Antes de solicitar uma exportação, siga as etapas de 1 a 3 em Tutorial de Java para criar um ledger de amostra e carregá-lo com dados de amostra.

O código do tutorial nas classes a seguir fornece exemplos de como criar uma exportação, verificar o status de uma exportação e processar a saída de uma exportação.

Classe	Descrição
ExportJournal	Exporta blocos de diário do ledger `vehicle-registration` de amostras para um intervalo de data e hora de 10 minutos atrás até agora. Grava os objetos de saída em um bucket S3 especificado ou cria um bucket exclusivo se não for fornecido.
DescribeJournalExport	Descreve um trabalho de exportação de diário para um item especificado `exportId` no ledger `vehicle-registration` de amostra.
ListJournalExports	Devolve uma lista de descrições de trabalhos de exportação de diários para o ledger `vehicle-registration` de amostra.
ValidateQldbHashChain	Valida a cadeia de hash do ledger de `vehicle-registration` amostra usando um determinado `exportId`. Se não for fornecido, solicita uma nova exportação para usar na validação da cadeia de hash.

Expiração do trabalho de exportação

Os trabalhos de exportação de diários concluídos estão sujeitos a um período de retenção de 7 dias. Eles são excluídos automaticamente após a expiração desse limite. Esse período de expiração é um limite fixo e não pode ser alterado.

Depois que um trabalho de exportação concluído for excluído, você não poderá mais usar o QLDB console ou as seguintes API operações para recuperar metadados sobre o trabalho:

- `DescribeJournalS3Export`
- `ListJournalS3Exports`
- `ListJournalS3ExportsForLedger`

No entanto, essa expiração não afeta os dados exportados em si. Todos os metadados são preservados nos arquivos de manifesto gravados por suas exportações. Essa expiração foi projetada para proporcionar uma experiência mais tranquila para as API operações que listam trabalhos de exportação de diários. QLDBremove trabalhos de exportação antigos para garantir que você veja apenas exportações recentes sem precisar analisar várias páginas de trabalhos.

Saída de exportação do diário em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Um trabalho de exportação de QLDB diário da Amazon grava dois arquivos de manifesto, além dos objetos de dados que contêm seus blocos de diário. Esses arquivos são salvos no bucket do Amazon S3 fornecido por você na solicitação de exportação. As seções a seguir descrevem o formato e o conteúdo de cada objeto de saída.

> **Note**
>
> Se você especificar JSON como formato de saída do seu trabalho de exportação, QLDB converterá negativamente os dados do diário Amazon Ion JSON em seus objetos de dados exportados. Para obter mais informações, acesse o Conversão descendente para JSON.

Tópicos

- Arquivos de manifesto
- Objetos de dados
- Conversão descendente para JSON
- Biblioteca de processadores de exportação (Java)

Arquivos de manifesto

QLDBA Amazon cria dois arquivos de manifesto no bucket S3 fornecido para cada solicitação de exportação. O arquivo de manifesto inicial é criado assim que você envia a solicitação de exportação. O arquivo de manifesto final é gravado após a conclusão da exportação. Você pode usar esses arquivos para verificar o status dos seus trabalhos de exportação no Amazon S3.

O formato do conteúdo dos arquivos de manifesto corresponde ao formato de saída solicitado para a exportação.

Manifesto inicial

O manifesto inicial indica que seu trabalho de exportação foi iniciado. Ele contém os parâmetros de entrada que você passou para a solicitação. Além do destino do Amazon S3 e dos parâmetros de horário de início e término da exportação, esse arquivo também contém um `exportId`. `exportId`É uma ID exclusiva que é QLDB atribuída a cada trabalho de exportação.

A convenção de nomenclatura de arquivos é a seguinte.

```
s3://amzn-s3-demo-bucket/prefix/exportId.started.manifest
```

Veja a seguir um exemplo de manifesto inicial e seu conteúdo no formato de texto Ion.

```
s3://amzn-s3-demo-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:"amzn-s3-demo-bucket",
  prefix:"journalExport",
  objectEncryptionType:"NO_ENCRYPTION",
  outputFormat:"ION_TEXT"
}
```

O manifesto inicial inclui `outputFormat` somente se tiver sido especificado na solicitação de exportação. Se você não especificar o formato de saída, os dados exportados assumirão o formato padrão `ION_TEXT`.

A API operação DescribeJournalS3Export e o tipo de conteúdo dos objetos Amazon S3 exportados também indicam o formato de saída.

Manifesto final

O manifesto final indica que seu trabalho de exportação para uma determinada cadeia do diário foi concluído. O trabalho de exportação grava um arquivo de manifesto final separado para cada cadeia.

> **Note**
>
> Na AmazonQLDB, um fio é uma partição do diário do seu livro contábil. QLDBatualmente suporta periódicos com apenas uma única vertente.

O manifesto final inclui uma lista ordenada de chaves de objetos de dados que foram gravadas durante a exportação. A convenção de nomenclatura de arquivos é a seguinte.

```
s3://amzn-s3-demo-bucket/prefix/exportId.strandId.completed.manifest
```

`strandId`É uma ID exclusiva que é QLDB atribuída à cadeia. Veja a seguir um exemplo de manifesto final e seu conteúdo em formato de texto Ion.

```
s3://amzn-s3-demo-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"
  ]
}
```

Objetos de dados

A Amazon QLDB grava objetos de dados de diário no bucket Amazon S3 fornecido na representação de texto ou binária do formato Amazon Ion ou no formato de texto JSONLines.

No formato JSON Linhas, cada bloco em um objeto de dados exportado é um JSON objeto válido delimitado por uma nova linha. Você pode usar esse formato para integrar diretamente JSON as exportações com ferramentas de análise, como Amazon Athena e AWS Glue porque esses serviços podem analisar automaticamente os delimitados por novas linhasJSON. Para obter mais informações sobre o formato, consulte JSONLinhas.

Nomes de objeto de dados

Um trabalho de exportação de diário grava esses objetos de dados com a seguinte convenção de nomenclatura.

```
s3://amzn-s3-demo-bucket/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- Os dados de saída de cada trabalho de exportação são divididos em partes.
- `yyyy/mm/dd/hh`: a data e a hora em que você enviou a solicitação de exportação. Objetos que são exportados na mesma hora são agrupados sob o mesmo prefixo do Amazon S3.
- `strandId`: o ID exclusivo da cadeia específica que contém o bloco de diário que está sendo exportado.
- `startSn-endSn`: o intervalo de números de sequência incluído no objeto. Um número de sequência especifica a localização de um bloco dentro de uma cadeia.

Por exemplo, suponha que você especifique o seguinte caminho.

```
s3://amzn-s3-demo-bucket/journalExport/
```

Seu trabalho de exportação cria um objeto de dados do Amazon S3 semelhante ao seguinte. Este exemplo mostra um nome de objeto no formato Ion.

```
s3://amzn-s3-demo-bucket/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

Conteúdo do objeto de dados

Cada objeto de dados contém objetos de bloco de diário com o seguinte formato.

```
{
  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
    }
  ]
}
```

Um bloco é um objeto que é confirmado no diário durante uma transação. Um bloco contém metadados da transação junto com entradas que representam as revisões do documento que foram confirmadas na transação e as declarações PartiQL que as confirmaram.

O exemplo a seguir é de um bloco com dados de amostra em formato de texto Ion. Para obter mais informações sobre os campos em um objeto de bloco, consulte Conteúdo do periódico na Amazon QLDB.

> **Note**
>
> Este exemplo de bloco é fornecido apenas para fins informativos. Os hashes mostrados não são valores reais de hash calculados.

```
{
  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"
      }
    }
  ]
}
```

No campo `revisions`, alguns objetos de revisão podem conter apenas um valor `hash` e nenhum outro atributo. Essas são revisões de sistema somente internas que não contêm dados do usuário. Um trabalho de exportação inclui essas revisões em seus respectivos blocos porque os hashes dessas revisões fazem parte da cadeia de hash completa do diário. A cadeia de hash completa é necessária para a verificação criptográfica.

Conversão descendente para JSON

Se você especificar JSON como formato de saída do seu trabalho de exportação, QLDB converterá negativamente os dados do diário Amazon Ion JSON em seus objetos de dados exportados. No entanto, a JSON conversão de íon em causa perdas em certos casos em que seus dados usam os tipos ricos de íons que não existem em. JSON

Para obter detalhes sobre as regras de JSON conversão de íons em, consulte Conversão descendente em JSON no Amazon Ion Cookbook.

Biblioteca de processadores de exportação (Java)

QLDBfornece uma estrutura extensível para Java que simplifica o processamento de exportações no Amazon S3. Essa biblioteca de estrutura lida com o trabalho de ler a saída de uma exportação e iterar os blocos exportados em ordem sequencial. Para usar esse processador de exportação, consulte o GitHub repositório amazon-qldb-export-processorawslabs/ -java.

Permissões de exportação de diário em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Antes de enviar uma solicitação de exportação de periódicos na AmazonQLDB, você deve QLDB fornecer permissões de gravação no bucket especificado do Amazon S3. Se você escolher um cliente gerenciado AWS KMS key como o tipo de criptografia de objeto para seu bucket do Amazon S3, você também deve QLDB fornecer permissões para usar sua chave de criptografia simétrica especificada. O Amazon S3 não oferece suporte a chaves KMSassimétricas.

Para fornecer ao seu trabalho de exportação as permissões necessárias, você pode QLDB assumir uma função de IAM serviço com as políticas de permissões apropriadas. Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.

> **Note**
>
> Para passar uma função QLDB ao solicitar a exportação de um diário, você deve ter permissões para realizar a `iam:PassRole` ação no recurso da IAM função. Isso é um acréscimo à `qldb:ExportJournalToS3` permissão no recurso QLDB contábil.
> Para saber como controlar o acesso ao QLDB usoIAM, consulteComo a Amazon QLDB trabalha com IAM. Para obter um exemplo QLDB de política, consulteExemplos de políticas baseadas em identidade para a Amazon QLDB.

Neste exemplo, você cria uma função que permite QLDB gravar objetos em um bucket do Amazon S3 em seu nome. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.

Se você estiver exportando um QLDB diário em seu Conta da AWS pela primeira vez, você deve primeiro criar uma IAM função com as políticas apropriadas fazendo o seguinte. Ou você pode usar o QLDB console para criar automaticamente a função para você. Caso contrário, você poderá escolher uma função que você criou anteriormente.

Tópicos

- Criação de uma política de permissões
- Crie uma IAM função

Criação de uma política de permissões

Conclua as etapas a seguir para criar uma política de permissões para um trabalho de exportação de QLDB diário. Este exemplo mostra uma política de bucket do Amazon S3 que concede QLDB permissões para gravar objetos em seu bucket especificado. Se aplicável, o exemplo também mostra uma política de chaves que permite QLDB usar sua KMS chave de criptografia simétrica.

Para obter mais informações sobre políticas de bucket do Amazon S3, consulte Uso de políticas de bucket e políticas de usuário no Guia do Usuário Amazon Simple Storage Service. Para saber mais a respeito AWS KMS políticas-chave, consulte Uso de políticas principais em AWS KMS no AWS Key Management Service Guia do desenvolvedor.

> **Note**
>
> Seu bucket e sua KMS chave do Amazon S3 devem estar no mesmo Região da AWS como seu QLDB livro contábil.

Para usar o editor JSON de políticas para criar uma política

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
2. Na coluna de navegação à esquerda, selecione Políticas.

   Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.
3. Na parte superior da página, escolha Criar política.
4. Escolha a JSONguia.
5. Insira um documento JSON de política.
   - Se você estiver usando uma KMS chave gerenciada pelo cliente para criptografia de objetos do Amazon S3, use o seguinte exemplo de documento de política. Para usar essa política, substitua *amzn-s3-demo-bucket*, *us-east-1*, *123456789012* e *1234abcd-12ab-34cd-56ef-1234567890ab* no exemplo com suas próprias informações.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        },
        {
            "Sid": "QLDBJournalExportKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

- Para outros tipos de criptografia, use o seguinte exemplo de documento de política. Para usar essa política, substitua *amzn-s3-demo-bucket* no exemplo com seu próprio nome de bucket do Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

6. Escolha Revisar política.

> **Note**
>
> Você pode alternar entre o editor visual e JSONas guias a qualquer momento. No entanto, se você fizer alterações ou escolher Revisar política na guia Editor visual, IAM poderá reestruturar sua política para otimizá-la para o editor visual. Para obter mais informações, consulte Reestruturação de políticas no Guia do IAM usuário.

7. Na página Review policy (Revisar política), insira um Name (Nome) e uma Description (Descrição) opcional para a política que você está criando. Revise o Resumo da política para ver as permissões que são concedidas pela política. Em seguida, escolha Criar política para salvar seu trabalho.

Crie uma IAM função

Depois de criar uma política de permissões para seu trabalho de exportação de QLDB periódicos, você pode criar uma IAM função e anexar sua política a ela.

Para criar a função de serviço para QLDB (IAMconsole)

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
2. No painel de navegação do IAM console, escolha Funções e, em seguida, escolha Criar função.
3. Para Tipo de entidade confiável, escolha AWS service (Serviço da AWS).
4. Para Serviço ou caso de uso, escolha e QLDB, em seguida, escolha o caso de QLDBuso.
5. Escolha Próximo.
6. Selecione a caixa ao lado da política que você criou nas etapas anteriores.
7. (Opcional) Defina um limite de permissões. Esse é um atributo avançado que está disponível para perfis de serviço, mas não para perfis vinculados ao serviço.

   a. Abra a seção Definir limite de permissões e escolha Usar um limite de permissões para controlar o número máximo de permissões do perfil.

      IAMinclui uma lista dos AWS políticas gerenciadas e gerenciadas pelo cliente em sua conta.

   b. Selecione a política a ser usada para o limite de permissões.
8. Escolha Próximo.

9. Insira um nome de perfil ou um sufixo de nome de perfil para ajudar a identificar a finalidade do perfil.

> **Important**
>
> Quando nomear um perfil, observe o seguinte:
>
> - Os nomes das funções devem ser exclusivos em seu Conta da AWS, e não pode ser tornada única caso a caso.
>
> Por exemplo, não crie dois perfis denominados **`PRODROLE`** e **`prodrole`**. Quando um nome de função é usado em uma política ou como parte de umaARN, o nome da função diferencia maiúsculas de minúsculas, no entanto, quando um nome de função aparece para os clientes no console, como durante o processo de login, o nome da função não diferencia maiúsculas de minúsculas.
> - Não é possível editar o nome do perfil depois de criá-lo porque outras entidades podem referenciar o perfil.

10. (Opcional) Em Descrição, insira uma descrição para o perfil.
11. (Opcional) Para editar os casos de uso e as permissões do perfil, escolha Editar nas seções Etapa 1: selecionar entidades confiáveis ou Etapa 2: adicionar permissões.
12. (Opcional) Para ajudar a identificar, organizar ou pesquisar o perfil, adicione tags como pares de chave-valor. Para obter mais informações sobre o uso de tags emIAM, consulte IAMRecursos de marcação no Guia do IAM usuário.
13. Reveja a função e escolha Criar função.

O JSON documento a seguir é um exemplo de uma política de confiança que QLDB permite assumir uma IAM função com permissões específicas anexadas a ela.

```
{
    "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**
>
> O exemplo a seguir mostra como é possível usar as chaves de contexto de condição globais `aws:SourceArn` e `aws:SourceAccount` para evitar o problema de substituto confuso. Com essa política de confiança, `123456789012` só QLDB pode assumir a função QLDB de qualquer recurso na conta.
> Para obter mais informações, consulte Prevenção do problema do substituto confuso entre serviços.

Depois de criar sua IAM função, retorne ao QLDB console e atualize a página Criar tarefa de exportação para que ele possa encontrar sua nova função.

Erros comuns na exportação de diários

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve os erros de tempo de execução que são gerados pela Amazon QLDB para solicitações de exportação de periódicos.

Veja a seguir uma lista de exceções comuns retornadas pelo serviço. Cada exceção inclui a mensagem de erro específica, seguida por uma breve descrição e sugestões de possíveis soluções.

AccessDeniedException

Mensagem: Usuário: *userARN* não está autorizado a executar: iam: PassRole no recurso: *roleARN*

Você não tem permissões para passar uma IAM função para o QLDB serviço. QLDBrequer uma função para todas as solicitações de exportação de periódicos, e você deve ter permissões para passar essa função paraQLDB. A função QLDB fornece permissões de gravação em seu bucket específico do Amazon S3.

Verifique se você define uma IAM política que conceda permissão para realizar a `PassRole` API operação no recurso de IAM função especificado para o QLDB serviço (`qldb.amazonaws.com`). Para ver um exemplo de política, consulte Exemplos de políticas baseadas em identidade para a Amazon QLDB.

IllegalArgumentException

Mensagem: QLDB encontrou um erro ao validar a configuração do S3: *errorCode errorMessage*

Uma possível causa desse erro é que o bucket do Amazon S3 fornecido não existe no Amazon S3. Ou QLDB não tem permissões suficientes para gravar objetos no bucket especificado do Amazon S3.

Verifique se o nome do bucket do S3 fornecido na solicitação de trabalho de exportação está correto. Para obter informações sobre a nomenclatura de buckets, consulte Restrições e limitações do bucket no Guia do usuário do Amazon Simple Storage Service.

Além disso, verifique se você define uma política para o bucket especificado que concede `PutObject` e `PutObjectAcl` autoriza o QLDB serviço (`qldb.amazonaws.com`). Para saber mais, consulte Permissões de exportação.

IllegalArgumentException

Mensagem: Resposta inesperada do Amazon S3 ao validar a configuração do S3. Resposta do S3: *errorCode errorMessage*

A tentativa de gravar dados de exportação do diário no bucket do S3 fornecido falhou com a resposta de erro fornecida pelo Amazon S3. Para obter mais informações sobre as causas possíveis, consulte Solução de problemas do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

IllegalArgumentException

Mensagem: O prefixo do bucket do Amazon S3 não deve exceder 128 caracteres

O prefixo fornecido na solicitação de exportação do diário contém mais de 128 caracteres.

IllegalArgumentException

Mensagem: A data de início não deve ser maior que a data de término

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de data e hora ISO8601 e no Tempo Universal Coordenado ()UTC.

IllegalArgumentException

Mensagem: A data de término não pode ser no futuro

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de `ISO 8601` data e hora e emUTC.

IllegalArgumentException

Mensagem: A configuração de criptografia de objeto fornecida (S3EncryptionConfiguration) não é compatível com um AWS Key Management Service (AWS KMS) chave

Você forneceu um `KMSKeyArn` com um `ObjectEncryptionType` de `NO_ENCRYPTION` ou `SSE_S3`. Você só pode fornecer um cliente gerenciado AWS KMS key para um tipo de criptografia de objeto de`SSE_KMS`. Para obter mais informações sobre o uso da criptografia no lado do servidor no Amazon S3, consulte Proteger dados usando criptografia no lado do servidor, no Guia do desenvolvedor do Amazon S3.

LimitExceededException

Mensagem: Excedeu o limite de 2 trabalhos de exportação de diário em execução simultânea

QLDBimpõe um limite padrão de dois trabalhos simultâneos de exportação de diário.

Streaming de dados de diários da Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB usa um registro transacional imutável, conhecido como diário, para armazenamento de dados. O diário rastreia todas as alterações em seus dados confirmados e mantém um histórico de alterações completo e verificável ao longo do tempo.

Você pode criar um stream QLDB que capture todas as revisões de documentos comprometidas com seu diário e entregue esses dados ao Amazon Kinesis Data Streams quase em tempo real. Um QLDB stream é um fluxo contínuo de dados do diário do seu livro contábil para um recurso de stream de dados do Kinesis.

Em seguida, você usa a plataforma de streaming Kinesis ou a Biblioteca de cliente Kinesis para consumir seu fluxo, processar os registros de dados e analisar o conteúdo dos dados. Um QLDB stream grava seus dados no Kinesis Data Streams em três tipos de registros: controle, resumo do bloco e detalhes da revisão. Para obter mais informações, consulte QLDBregistros de streaming no Kinesis.

Tópicos

- Casos de uso comuns
- Consumindo seu fluxo
- Garantia de entrega
- Considerações sobre latência de entrega
- Introdução aos fluxos
- Criação e gerenciamento de fluxos em QLDB
- Desenvolvendo com fluxos em QLDB
- QLDBregistros de streaming no Kinesis
- Permissões de transmissão em QLDB
- Erros comuns em fluxos de diários em QLDB

Casos de uso comuns

O streaming permite que você o use QLDB como uma fonte única e verificável de verdade, ao mesmo tempo em que integra os dados do seu diário a outros serviços. A seguir estão alguns dos casos de uso comuns suportados por fluxos de QLDB diários:

- Arquitetura orientada a eventos: crie aplicativos em um estilo arquitetônico orientado por eventos com componentes desacoplados. Por exemplo, um banco pode usar AWS Lambda funções para implementar um sistema de notificação que alerta os clientes quando o saldo da conta cai abaixo de um limite. Nesse sistema, os saldos das contas são mantidos em um QLDB livro contábil e quaisquer alterações no saldo são registradas no diário. A ferramenta AWS Lambda a função pode acionar a lógica de notificação ao consumir um evento de atualização de saldo que é confirmado no diário e enviado para um stream de dados do Kinesis.
- Análise em tempo real: crie aplicativos de consumo da Kinesis que executam análises em tempo real sobre dados de eventos. Com esse recurso, você pode obter insights quase em tempo real e responder rapidamente a um ambiente de negócios em constante mudança. Por exemplo, um site de comércio eletrônico pode analisar dados de vendas de produtos e interromper os anúncios de um produto com desconto assim que as vendas atingirem um limite.
- Análise histórica — Aproveite a arquitetura orientada a periódicos da Amazon QLDB reproduzindo dados históricos de eventos. Você pode optar por iniciar um QLDB stream a partir de qualquer momento no passado, no qual todas as revisões feitas desde aquela época são entregues ao Kinesis Data Streams. Usando esse atributo, você pode criar aplicativos de consumo do Kinesis que executam trabalhos de análise em dados históricos. Por exemplo, um site de comércio eletrônico pode executar análises conforme necessário para gerar métricas de vendas anteriores que não foram capturadas anteriormente.
- Replicação para bancos de dados específicos — Conecte livros contábeis a outros QLDB armazenamentos de dados específicos usando fluxos de diários. QLDB Por exemplo, use a plataforma de dados de streaming Kinesis para se integrar ao Amazon OpenSearch Service, que pode fornecer recursos de pesquisa de texto completo para QLDB documentos. Você também pode criar aplicativos de consumo personalizados do Kinesis para replicar os dados do diário em outros bancos de dados com propósito específico que fornecem diferentes visões materializadas. Por exemplo, replique para o Amazon Aurora para obter dados relacionais ou para o Amazon Neptune para dados baseados em gráficos.

Consumindo seu fluxo

Use o Kinesis Data Streams para consumir, processar e analisar continuamente grandes fluxos de registros de dados. Além do Kinesis Data Streams, a plataforma de streaming de dados Kinesis inclui o Amazon Data Firehose e o Amazon Managed Service para Apache Flink. Você pode usar essa plataforma para enviar registros de dados diretamente para serviços como Amazon OpenSearch Service, Amazon Redshift, Amazon S3 ou Splunk. Para obter mais informações, consulte Consumidores Kinesis Data Streams no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Você também pode usar a Kinesis Client Library (KCL) para criar um aplicativo consumidor de stream para processar registros de dados de forma personalizada. Isso KCL simplifica a codificação ao fornecer abstrações úteis acima do Kinesis Data Streams de baixo nível. API Para saber mais sobre oKCL, consulte Como usar a biblioteca de cliente do Kinesis no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Garantia de entrega

QLDBos streams fornecem uma garantia at-least-oncede entrega. Cada registro de dados produzido por um QLDB stream é entregue ao Kinesis Data Streams pelo menos uma vez. Os mesmos registros podem aparecer em um fluxo de dados do Kinesis várias vezes. Portanto, você deve ter uma lógica de desduplicação na camada de aplicação do consumidor, se o seu caso de uso assim o exigir.

Também não há garantias de pedido. Em algumas circunstâncias, QLDB blocos e revisões podem ser produzidos em um stream de dados do Kinesis fora de ordem. Para obter mais informações, consulte Lidando com duplicatas e registros out-of-order.

Considerações sobre latência de entrega

QLDBos streams normalmente entregam atualizações para o Kinesis Data Streams quase em tempo real. No entanto, os cenários a seguir podem criar latência adicional antes que QLDB os dados recém-confirmados sejam emitidos para um stream de dados do Kinesis:

- O Kinesis pode controlar os dados que são transmitidosQLDB, dependendo do provisionamento do Kinesis Data Streams. Por exemplo, isso pode ocorrer se você tiver vários QLDB streams gravando em um único stream de dados do Kinesis e a taxa QLDB de solicitação exceder a capacidade do recurso de stream do Kinesis. O controle de utilização no Kinesis também pode

ocorrer ao usar o provisionamento sob demanda se o throughput aumentar para mais que o dobro do pico anterior em menos de 15 minutos.

Você pode medir esse throughput excedido monitorando a métrica do Kinesis `WriteProvisionedThroughputExceeded`. Para obter mais informações e possíveis soluções, consulte Como soluciono erros de controle de utilização no Kinesis Data Streams?.

- Com QLDB os streams, você pode criar um stream indefinido com data e hora de início no passado e sem data e hora de término. Por design, QLDB começa a emitir dados recém-confirmados para o Kinesis Data Streams somente depois que todos os dados anteriores da data e hora de início especificadas forem entregues com sucesso. Se você perceber latência adicional nesse cenário, talvez seja necessário aguardar a entrega dos dados anteriores ou iniciar a transmissão a partir de uma data e hora de início posteriores.

Introdução aos fluxos

Veja a seguir uma visão geral de alto nível das etapas necessárias para começar a fazer fluxo de dados do diário para o Kinesis Data Streams:

1. Crie um recurso do Kinesis Data Streams. Para obter instruções, consulte Criação e atualização de fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.
2. Crie uma IAM função que permita QLDB assumir permissões de gravação para o stream de dados do Kinesis. Para obter instruções, consulte Permissões de transmissão em QLDB.
3. Crie um stream de QLDB diário. Para obter instruções, consulte Criação e gerenciamento de fluxos em QLDB.
4. Consuma o fluxo de dados do Kinesis, conforme descrito na seção anterior Consumindo seu fluxo. Para exemplos de código que mostram como usar a Biblioteca de Cliente Kinesis ou AWS Lambda, consulte Desenvolvendo com fluxos em QLDB.

Criação e gerenciamento de fluxos em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon fornece API operações para criar e gerenciar um fluxo de dados de diário do seu livro contábil para o Amazon Kinesis Data Streams. O QLDB stream captura todas as revisões de documentos enviadas ao seu diário e as envia para um stream de dados do Kinesis.

Você pode usar o AWS Management Console, um AWS SDK, ou o AWS Command Line Interface (AWS CLI) para criar um fluxo de diário. Além disso, você também pode usar um AWS CloudFormationmodelo para criar fluxos. Para obter mais informações, consulte o recurso AWS:QLDB:: :Stream no AWS CloudFormation Guia do usuário.

Tópicos

- Parâmetros de fluxo
- Transmitir ARN
- AWS Management Console
- Estados de fluxo
- Tratamento de fluxos prejudicados

Parâmetros de fluxo

Para criar um stream de QLDB diário, você deve fornecer os seguintes parâmetros de configuração:

Nome do ledger

O QLDB livro contábil cujos dados do diário você deseja transmitir para o Kinesis Data Streams.

Nome do fluxo

O nome que você deseja atribuir ao fluxo do QLDB diário. Os nomes definidos pelo usuário podem ajudar a identificar e indicar a finalidade de um stream.

O nome do fluxo deve ser exclusivo entre outros fluxos ativos em um ledger. Os nomes de fluxos têm as mesmas restrições de nomenclatura que os nomes do ledger, conforme definido em Cotas e limites na Amazon QLDB.

Além do nome do stream, QLDB atribui um ID de stream a cada QLDB stream que você cria. O ID do fluxo é exclusivo entre todos os fluxos de um determinado ledger, independentemente de seu status.

Data e hora de início

A data e hora de início a partir das quais iniciar o streaming de dados do diário. Esse valor pode ser qualquer data e hora no passado, mas não no futuro.

Data e hora de encerramento

(Opcional) A data e hora exclusivas que especificam quando o fluxo termina.

Se você criar um fluxo indefinido sem hora de término, deverá cancelá-lo manualmente para finalizar o fluxo. Você também pode cancelar um fluxo ativo e finito que ainda não tenha atingido a data e a hora de término especificadas.

Fluxo de dados do Kinesis de destino

O recurso de destino do Kinesis Data Streams no qual seu fluxo grava os registros de dados. Para saber como criar um fluxo de dados do Kinesis, consulte Criação e atualização de fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

> **Important**
>
> - Não há suporte para streams entre contas e entre regiões. O stream de dados do Kinesis especificado deve estar no mesmo Região da AWS e conta como seu livro contábil.
> - A agregação de registros no Kinesis Data Streams está ativada por padrão. Essa opção permite QLDB publicar vários registros de dados em um único registro do Kinesis Data Streams, aumentando o número de registros API enviados por chamada.
>
> A agregação de registros tem implicações importantes para o processamento de registros e requer desagregação em seu consumidor de stream. Para saber mais, consulte os KPL principais conceitos e a desagregação de consumidores no Guia do desenvolvedor do Amazon Kinesis Data Streams.

IAMpapel

A IAM função que permite QLDB assumir permissões de gravação em seu stream de dados do Kinesis. Você pode usar o QLDB console para criar automaticamente essa função ou criá-la manualmente noIAM. Para saber como criá-la manualmente, consulte Permissões de fluxo.

Para passar uma função QLDB ao solicitar um stream de diário, você deve ter permissões para realizar a `iam:PassRole` ação no recurso de IAM função.

Transmitir ARN

Cada stream de QLDB diário é um sub-recurso de um livro contábil e é identificado exclusivamente por um nome de recurso da Amazon (). ARN Veja a seguir um exemplo ARN de um QLDB fluxo com um ID de fluxo `IiPT4brpZCqCq3f4MTHbYy` para um livro contábil chamado`exampleLedger`.

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

A seção a seguir descreve como criar e cancelar um QLDB stream usando o AWS Management Console.

AWS Management Console

Siga estas etapas para criar ou cancelar uma QLDB transmissão usando o QLDB console.

Para criar um fluxo (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. Selecione Streams (Fluxos) no painel de navegação.
3. Escolha Create QLDB stream (Criar streaming do AK).
4. Na página Criar QLDB transmissão, insira as seguintes configurações:

   - Nome do fluxo — O nome que você deseja atribuir ao QLDB fluxo.
   - ledger: o ledger cujos dados do diário você deseja transmitir.
   - Data e hora de início — O carimbo de data/hora inclusivo no Tempo Universal Coordenado (UTC) a partir do qual iniciar a transmissão dos dados do diário. O padrão desse timestamp é a data e hora atual. Não pode ser no futuro e deve ser anterior à data e hora de término.
   - Data e hora de término — (Opcional) O carimbo de data/hora exclusivo (UTC) que especifica quando o stream termina. Se você deixar esse parâmetro em branco, o fluxo será executado indefinidamente até ser cancelado.
   - Stream de destino: o recurso de destino do Kinesis Data Streams no qual seu fluxo grava os registros de dados. Use o ARN formato a seguir.

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     Veja um exemplo a seguir.

```
arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
```

Não há suporte para streams entre contas e entre regiões. O stream de dados do Kinesis especificado deve estar no mesmo Região da AWS e conta como seu livro contábil.

- Ativar a agregação de registros no Kinesis Data Streams — (ativado por padrão) QLDB Permite publicar vários registros de dados em um único registro do Kinesis Data Streams, aumentando o número de registros enviados por chamada. API
- Acesso ao serviço — A IAM função que concede permissões de QLDB gravação ao seu stream de dados do Kinesis.

  Para passar uma função QLDB ao solicitar um stream de diário, você deve ter permissões para realizar a `iam:PassRole` ação no recurso de IAM função.

  - Crie e use um novo perfil de serviço: deixe o console criar uma nova função para você com as permissões necessárias para o fluxo de dados do Kinesis especificado.
  - Use uma função de serviço existente — Para saber como criar essa função manualmente emIAM, consultePermissões de fluxo.
- Tags: (opcional) adicione metadados ao fluxo anexando tags como pares de chave-valor. Você pode adicionar tags ao seu fluxo para ajudar a organizar e identificá-los. Para obter mais informações, consulte Marcação de recursos da Amazon QLDB.

  Escolha Adicionar tag e, em seguida, insira os pares de valores-chave, conforme apropriado.

5. Quando as configurações estiverem como você deseja, escolha Criar QLDB transmissão.

   Se o envio da solicitação for bem-sucedido, o console retornará à página principal do Streams e listará seus QLDB streams com o status atual.

6. Depois que seu fluxo estiver ativo, use o Kinesis para processar seus dados de fluxo com um aplicativo de consumidor.

   Abra o console do Kinesis Data Streams https://console.aws.amazon.com/kinesis/em.

Para obter mais informações sobre o formato dos registros de dados de fluxo, consulte QLDBregistros de streaming no Kinesis.

Para saber como lidar com fluxos que resultam em erro, consulte Tratamento de fluxos prejudicados.

Para cancelar um fluxo (console)

Você não pode reiniciar uma QLDB transmissão depois de cancelá-la. Para retomar a entrega de seus dados para o Kinesis Data Streams, você pode criar um QLDB novo stream.

1. Abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. Selecione Streams (Fluxos) no painel de navegação.
3. Na lista de QLDB streams, selecione o stream ativo que você deseja cancelar.
4. Escolha Cancelar fluxo. Confirme isso inserindo **`cancel stream`** na caixa fornecida.

Para obter informações sobre como usar o QLDB API com um AWS SDKou o AWS CLI para criar e gerenciar fluxos de diário, consulteDesenvolvendo com fluxos em QLDB.

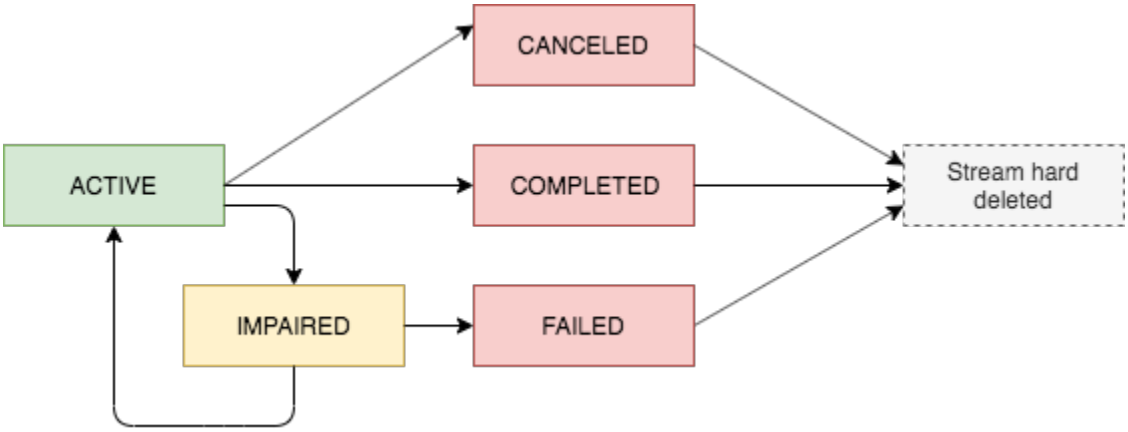
Estados de fluxo

O status de um QLDB stream pode ser um dos seguintes:

- `ACTIVE`: no momento, está transmitindo ou aguardando para transmitir dados (para um fluxo indefinido sem hora de término).
- `COMPLETED`: concluiu com sucesso o streaming de todos os blocos de diário dentro do intervalo de tempo especificado. Este é um estado terminal.
- `CANCELED`: foi encerrado por uma solicitação do usuário antes do horário de término especificado e não está mais transmitindo dados ativamente. Este é um estado terminal.
- `IMPAIRED`: não é possível gravar registros no Kinesis devido a um erro que exige sua ação. Esse é um estado recuperável e não terminal.

  Se você resolver o erro em uma hora, o stream será automaticamente transferido para o estado `ACTIVE`. Se o erro continuar sem solução após uma hora, o stream passa automaticamente para o estado `FAILED`.
- `FAILED`: não consegue gravar registros no Kinesis devido a um erro e está em um estado terminal irrecuperável.

O diagrama a seguir ilustra como um recurso de QLDB fluxo pode fazer a transição entre estados.



Expiração para fluxos terminais

Os recursos de fluxo que estão em um estado terminal (`CANCELED`, `COMPLETED`, e `FAILED`) estão sujeitos a um período de retenção de 7 dias. Eles são excluídos automaticamente após a expiração desse limite.

Depois que um stream de terminal é excluído, você não pode mais usar o QLDB console ou o QLDB API para descrever ou listar o recurso de stream.

Tratamento de fluxos prejudicados

Se seu fluxo encontrar um erro, ele será movido para o estado `IMPAIRED` primeiro. QLDBcontinua a repetir as `IMPAIRED` transmissões por até uma hora.

Se você resolver o erro em uma hora, o stream será automaticamente transferido para o estado `ACTIVE`. Se o erro continuar sem solução após uma hora, o stream passa automaticamente para o estado `FAILED`.

Um fluxo danificado ou com falha pode ser uma das seguintes causas de erro:

- `KINESIS_STREAM_NOT_FOUND`: o recurso Kinesis Data Streams de destino não existe. Verifique se o stream de dados do Kinesis que você forneceu na sua solicitação de QLDB stream está correto. Em seguida, acesse Kinesis e crie o fluxo de dados que você especificou.
- `IAM_PERMISSION_REVOKED`— QLDB não tem permissões suficientes para gravar registros de dados em seu stream de dados do Kinesis especificado. Verifique se você define uma política para o stream de dados do Kinesis especificado que conceda ao QLDB service (`qldb.amazonaws.com`) permissões para as seguintes ações:
  - `kinesis:PutRecord`

- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

Monitorar os fluxos prejudicados

Se uma transmissão for prejudicada, o QLDB console exibirá um banner que mostra detalhes sobre a transmissão e o erro que ela encontrou. Você também pode usar a `DescribeJournalKinesisStream` API operação para obter o status de um stream e a causa do erro subjacente.

Além disso, você pode usar CloudWatch a Amazon para criar um alarme que monitora a `IsImpaired` métrica de um stream. Para obter informações sobre o monitoramento de QLDB métricas com CloudWatch, consulteQLDBDimensões e métricas da Amazon.

Desenvolvendo com fluxos em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção resume as API operações que você pode usar com um AWS SDKou o AWS CLI para criar e gerenciar fluxos de periódicos na AmazonQLDB. Também descreve os aplicativos de amostra que demonstram essas operações e usam a Biblioteca de Cliente Kinesis () KCL ou AWS Lambda para implementar um consumidor de stream.

Você pode usar o KCL para criar aplicativos de consumo para o Amazon Kinesis Data Streams. Isso KCL simplifica a codificação ao fornecer abstrações úteis acima do Kinesis Data Streams de baixo nível. API Para saber mais sobre oKCL, consulte Como usar a biblioteca de cliente do Kinesis no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Sumário

- QLDBfluxo de diário APIs

- Aplicações de exemplo
  - Operações básicas (Java)
  - Integração com o OpenSearch serviço (Python)
  - Integração com Amazon SNS e Amazon SQS (Python)

QLDBfluxo de diário APIs

O QLDB API fornece as seguintes operações de fluxo de diário para uso por programas aplicativos:

- `StreamJournalToKinesis`— Cria um fluxo de diário para um determinado QLDB livro contábil. O fluxo captura todas as revisões de documentos confirmadas no diário do ledger e entrega os dados a um recurso especificado do Amazon Kinesis Data Streams.
  - A agregação de registros no Kinesis Data Streams está ativada por padrão. Essa opção permite QLDB publicar vários registros de dados em um único registro do Kinesis Data Streams, aumentando o número de registros API enviados por chamada.

    A agregação de registros tem implicações importantes para o processamento de registros e requer desagregação em seu consumidor de stream. Para saber mais, consulte os KPL principais conceitos e a desagregação de consumidores no Guia do desenvolvedor do Amazon Kinesis Data Streams.
- `DescribeJournalKinesisStream`— Retorna informações detalhadas sobre um determinado fluxo de QLDB diário. A saída inclui o nome do streamARN, o status atual, a hora da criação e os parâmetros da sua solicitação original de criação do stream.
- `ListJournalKinesisStreamsForLedger`— Retorna uma lista de todos os descritores de fluxo de QLDB diário para um determinado livro contábil. A saída de cada descritor de fluxo inclui os mesmos detalhes devolvidos por `DescribeJournalKinesisStream`.
- `CancelJournalKinesisStream`— Encerra um determinado fluxo de QLDB diário. Antes que um fluxo possa ser cancelado, seu status atual deve ser `ACTIVE`.

  Não é possível reiniciar um fluxo depois de tê-lo cancelado. Para retomar a entrega de seus dados para o Kinesis Data Streams, você pode criar um QLDB novo stream.

Para obter descrições completas dessas API operações, consulte QLDBAPIReferência da Amazon o.

Para obter informações sobre como criar e gerenciar fluxos de diários usando o AWS CLI, veja o AWS CLI Referência de comando.

Aplicações de exemplo

QLDBfornece exemplos de aplicativos que demonstram várias operações usando fluxos de diários. Esses aplicativos são de código aberto no AWS GitHub Site de amostras.

Tópicos

- Operações básicas (Java)
- Integração com o OpenSearch serviço (Python)
- Integração com Amazon SNS e Amazon SQS (Python)

Operações básicas (Java)

Para ver um exemplo de código Java que demonstra operações básicas para fluxos de QLDB diários, consulte o GitHub repositório aws-samples/ -java. amazon-qldb-dmv-sample Para obter instruções sobre como baixar e instalar esse aplicativo de amostra, consulte Instalando o aplicativo de amostra Amazon QLDB Java.

> **Note**
>
> Depois de instalar o aplicativo, não vá para a Etapa 1 do tutorial Java para criar um ledger. Este aplicativo de amostra para streaming cria o ledger `vehicle-registration` para você.

Esse aplicativo de amostra empacota o código-fonte completo do Tutorial de Java e de suas dependências, incluindo os seguintes módulos:

- AWS SDK for Java— Criar e excluir os recursos do Kinesis Data Streams QLDB e do Kinesis, incluindo QLDB livros contábeis, fluxos de diários e fluxos de dados do Kinesis.
- QLDBDriver Amazon para Java: executar transações de dados em um ledger usando instruções PartiQL, incluindo a criação de tabelas e a inserção de documentos.
- Kinesis Client Library: consumir e processar dados de um fluxo de dados do Kinesis.

Executar o código

A StreamJournalclasse contém um código tutorial que demonstra as seguintes operações:

1. Crie um ledger chamado `vehicle-registration`, crie tabelas e carregue-as com dados de amostra.

   > **Note**
   >
   > Antes de executar esse código, confirme que você ainda não tem um ledger ativo chamado `vehicle-registration`.

2. Crie um stream de dados do Kinesis, uma IAM função que permite QLDB assumir permissões de gravação para o stream de dados do Kinesis e um stream de diário. QLDB
3. Use o KCL para iniciar um leitor de stream que processa o stream de dados do Kinesis e registra cada registro de QLDB dados.
4. Use os dados do fluxo para validar a cadeia de hash do ledger `vehicle-registration` de amostras.
5. Limpe todos os recursos interrompendo o leitor de stream, cancelando o stream do QLDB diário, excluindo o livro contábil e excluindo o stream de dados do Kinesis.

Para executar o código do tutorial `StreamJournal`, insira o seguinte comando do Gradle no diretório raiz do seu projeto.

```
./gradlew run -Dtutorial=streams.StreamJournal
```

Integração com o OpenSearch serviço (Python)

Para ver um aplicativo de amostra em Python que demonstra como integrar um stream QLDB com o Amazon OpenSearch Service, consulte o GitHub repositório aws-samples/ -. amazon-qldb-streaming-amazon opensearch-service-sample-python Este aplicativo usa um AWS Lambda função para implementar um consumidor do Kinesis Data Streams.

Para clonar o repositório, digite o seguinte comando `git`.

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-
service-sample-python.git
```

Para executar o aplicativo de amostra, consulte o READMEon GitHub para obter instruções.

Integração com Amazon SNS e Amazon SQS (Python)

Para ver um aplicativo de amostra em Python que demonstra como integrar um stream QLDB com o Amazon Simple Notification Service (AmazonSNS), consulte o GitHub repositório aws-samples/ -. amazon-qldb-streams-dmv sample-lambda-python

Este aplicativo usa um AWS Lambda função para implementar um consumidor do Kinesis Data Streams. Ele envia mensagens para um SNS tópico da Amazon, que tem uma fila do Amazon Simple Queue Service (AmazonSQS) inscrita nele.

Para clonar o repositório, digite o seguinte comando `git`.

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-
python.git
```

Para executar o aplicativo de amostra, consulte o READMEon GitHub para obter instruções.

QLDBregistros de streaming no Kinesis

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Um QLDB stream da Amazon grava três tipos de registros de dados em um determinado recurso do Amazon Kinesis Data Streams: controle, resumo do bloco e detalhes da revisão. Todos os três tipos de registro são escritos na representação binária do formato Amazon Ion.

Os registros de controle indicam o início e a conclusão de suas QLDB transmissões. Sempre que uma revisão é confirmada em seu diário, um QLDB stream grava todos os dados do bloco de diário associado nos registros de resumo do bloco e detalhes da revisão.

Os três tipos de registro são polimórficos. Todos eles consistem em um registro comum de nível superior que contém o QLDB fluxoARN, o tipo de registro e a carga útil do registro. Esse registro de nível superior tem o formato a seguir.

```
{
  qldbStreamArn: string,
```

```
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

O campo `recordType` pode ter um dos três valores:

- `CONTROL`
- `BLOCK_SUMMARY`
- `REVISION_DETAILS`

As seções a seguir descrevem o formato e o conteúdo de cada registro de carga útil individual.

> **Note**
>
> QLDBgrava todos os registros de stream no Kinesis Data Streams na representação binária do Amazon Ion. Os exemplos a seguir são fornecidos na representação de texto do Ion para ilustrar o conteúdo do registro em um formato legível.

Tópicos

- Registros de controle
- Bloquear registros resumidos
- Registros de detalhes da revisão
- Lidando com duplicatas e registros out-of-order

Registros de controle

Um QLDB stream grava registros de controle para indicar seus eventos de início e conclusão. A seguir estão exemplos de registros de controle com dados de amostra para cada `controlRecordType`:

- `CREATED`— O primeiro registro que um QLDB stream grava no Kinesis para indicar que seu stream recém-criado está ativo.

```
{
```

```
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"CREATED"
  }
}
```

- `COMPLETED`— O último registro que um QLDB stream grava no Kinesis para indicar que seu stream atingiu a data e a hora de término especificadas. Esse registro não será gravado se você cancelar a transmissão.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

Bloquear registros resumidos

Um registro de resumo em bloco representa um bloco de diário no qual suas revisões de documentos são confirmadas. Um bloco é um objeto que é enviado ao seu QLDB diário durante uma transação.

A carga útil de um registro de resumo do bloco contém o endereço do bloco, a data e hora e outros metadados da transação que confirmou o bloqueio. Também inclui atributos resumidos das revisões no bloco e das declarações partiQL que as confirmaram. Veja a seguir um exemplo de um registro de resumo de bloco com dados de amostra.

> **Note**
>
> Este exemplo de resumo de bloco é fornecido apenas para fins informativos. Os hashes mostrados não são valores reais de hash calculados.

```
{
```

```
  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"
      }
    ]
  }
}
```

No campo `revisionSummaries`, algumas revisões podem não ter um `documentId`. Essas são revisões de sistema somente internas que não contêm dados do usuário. Um QLDB stream inclui essas revisões em seus respectivos registros de resumo de blocos porque os hashes dessas revisões fazem parte da cadeia de hash completa do periódico. A cadeia de hash completa é necessária para a verificação criptográfica.

Somente as revisões que têm um ID de documento são publicadas em registros separados de detalhes da revisão, conforme descrito na seção a seguir.

Registros de detalhes da revisão

Um registro de detalhes da revisão representa uma revisão do documento que está comprometida com seu diário. A carga contém todos os atributos da visualização confirmada da revisão, junto com o nome da tabela associada e o ID da tabela. Veja a seguir um exemplo de um registro de revisão com dados de amostra.

```
{
  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"
      }
    }
  }
}
```

Lidando com duplicatas e registros out-of-order

QLDBos streams podem publicar duplicatas e out-of-order registros no Kinesis Data Streams. Portanto, um aplicativo consumidor pode precisar implementar sua própria lógica para identificar e lidar com esses cenários. O resumo do bloco e os registros de detalhes da revisão incluem campos que você pode usar para essa finalidade. Combinados com os atributos dos serviços posteriores, esses campos podem indicar tanto uma identidade exclusiva quanto uma ordem estrita para os registros.

Por exemplo, considere um fluxo que se integra a QLDB um OpenSearch índice para fornecer recursos de pesquisa de texto completo em documentos. Nesse caso de uso, você precisa evitar

a indexação de revisões obsoletas (out-of-order) de um documento. Para impor o pedido e a desduplicação, você pode usar os campos ID do documento e versão externa em OpenSearch, junto com os campos ID do documento e versão em um registro de detalhes da revisão.

Para ver um exemplo de lógica de desduplicação em um aplicativo de amostra que se integra ao QLDB Amazon OpenSearch Service, consulte o GitHub repositório aws-samples/ -. amazon-qldb-streaming-amazon opensearch-service-sample-python

Permissões de transmissão em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Antes de criar um QLDB stream da Amazon, você deve QLDB fornecer permissões de gravação para seu recurso específico do Amazon Kinesis Data Streams. Se você estiver usando um serviço gerenciado pelo cliente AWS KMS key para a criptografia do lado do servidor do seu stream do Kinesis, você também deve QLDB fornecer permissões para usar sua chave de criptografia simétrica especificada. O Kinesis Data Streams não KMS é compatível com chaves assimétricas.

Para fornecer ao seu QLDB stream as permissões necessárias, você pode QLDB assumir uma função de IAM serviço com as políticas de permissões apropriadas. Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.

> **Note**
>
> Para passar uma função QLDB ao solicitar um stream de diário, você deve ter permissões para realizar a `iam:PassRole` ação no recurso de IAM função. Isso é um acréscimo à `qldb:StreamJournalToKinesis` permissão no sub-recurso do QLDB stream.

Para saber como controlar o acesso ao QLDB usoIAM, consulteComo a Amazon QLDB trabalha com IAM. Para obter um exemplo QLDB de política, consulteExemplos de políticas baseadas em identidade para a Amazon QLDB.

Neste exemplo, você cria uma função que permite QLDB gravar registros de dados em um stream de dados do Kinesis em seu nome. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.

Se você estiver transmitindo um QLDB diário em seu Conta da AWS pela primeira vez, você deve primeiro criar uma IAM função com as políticas apropriadas fazendo o seguinte. Ou você pode usar o QLDB console para criar automaticamente a função para você. Caso contrário, você poderá escolher uma função que você criou anteriormente.

Tópicos

- Criação de uma política de permissões
- Crie uma IAM função

Criação de uma política de permissões

Conclua as etapas a seguir para criar uma política de permissões para um QLDB stream. Este exemplo mostra uma política do Kinesis Data Streams QLDB que concede permissões para gravar registros de dados no stream de dados especificado do Kinesis. Se aplicável, o exemplo também mostra uma política de chaves que permite QLDB usar sua KMS chave de criptografia simétrica.

Para obter mais informações sobre as políticas do Kinesis Data Streams, consulte Controle do acesso aos recursos do Amazon Kinesis Data IAM Streams usando e Permissões KMS para usar chaves geradas pelo usuário no Guia do desenvolvedor do Amazon Kinesis Data Streams. Para saber mais a respeito AWS KMS políticas-chave, consulte Uso de políticas principais em AWS KMS no AWS Key Management Service Guia do desenvolvedor.

> **Note**
>
> Seu stream de dados e sua KMS chave do Kinesis devem estar no mesmo Região da AWS e conta como seu QLDB livro contábil.

Para usar o editor JSON de políticas para criar uma política

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
2. Na coluna de navegação à esquerda, selecione Políticas.

   Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.
3. Na parte superior da página, escolha Criar política.
4. Escolha a JSONguia.
5. Insira um documento JSON de política.
   - Se você estiver usando uma KMS chave gerenciada pelo cliente para a criptografia do lado do servidor do seu stream do Kinesis, use o seguinte exemplo de documento de política. Para usar essa política, substitua *us-east-1*, *123456789012*, *kinesis-stream-name* e *1234abcd-12ab-34cd-56ef-1234567890ab* no exemplo com suas próprias informações.

```
{
    "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"
        }
    ]
}
```

- Ou, use o seguinte exemplo de documento de política. Para usar essa política, substitua *us-east-1*, *123456789012* e *kinesis-stream-name* no exemplo com suas próprias informações.

```
{
    "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. Escolha Revisar política.

> **Note**
>
> Você pode alternar entre o editor visual e JSONas guias a qualquer momento. No entanto, se você fizer alterações ou escolher Revisar política na guia Editor visual, IAM poderá reestruturar sua política para otimizá-la para o editor visual. Para obter mais informações, consulte Reestruturação de políticas no Guia do IAM usuário.

7. Na página Review policy (Revisar política), insira um Name (Nome) e uma Description (Descrição) opcional para a política que você está criando. Revise o Resumo da política para ver as permissões que são concedidas pela política. Em seguida, escolha Criar política para salvar seu trabalho.

Crie uma IAM função

Depois de criar uma política de permissões para seu QLDB stream, você pode criar uma IAM função e anexar sua política a ela.

Para criar a função de serviço para QLDB (IAMconsole)

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
2. No painel de navegação do IAM console, escolha Funções e, em seguida, escolha Criar função.
3. Para Tipo de entidade confiável, escolha AWS service (Serviço da AWS).
4. Para Serviço ou caso de uso, escolha e QLDB, em seguida, escolha o caso de QLDBuso.
5. Escolha Próximo.
6. Selecione a caixa ao lado da política que você criou nas etapas anteriores.
7. (Opcional) Defina um limite de permissões. Esse é um atributo avançado que está disponível para perfis de serviço, mas não para perfis vinculados ao serviço.

   a. Abra a seção Definir limite de permissões e escolha Usar um limite de permissões para controlar o número máximo de permissões do perfil.

      IAMinclui uma lista dos AWS políticas gerenciadas e gerenciadas pelo cliente em sua conta.

   b. Selecione a política a ser usada para o limite de permissões.
8. Escolha Próximo.
9. Insira um nome de perfil ou um sufixo de nome de perfil para ajudar a identificar a finalidade do perfil.

   > **Important**
   >
   > Quando nomear um perfil, observe o seguinte:
   >
   > - Os nomes das funções devem ser exclusivos em seu Conta da AWS, e não pode ser tornada única caso a caso.
   >
   > Por exemplo, não crie dois perfis denominados **`PRODROLE`** e **`prodrole`**. Quando um nome de função é usado em uma política ou como parte de umaARN, o nome da função diferencia maiúsculas de minúsculas, no entanto, quando um nome de função aparece para os clientes no console, como durante o processo de login, o nome da função não diferencia maiúsculas de minúsculas.
   > - Não é possível editar o nome do perfil depois de criá-lo porque outras entidades podem referenciar o perfil.
10. (Opcional) Em Descrição, insira uma descrição para o perfil.

11. (Opcional) Para editar os casos de uso e as permissões do perfil, escolha Editar nas seções Etapa 1: selecionar entidades confiáveis ou Etapa 2: adicionar permissões.
12. (Opcional) Para ajudar a identificar, organizar ou pesquisar o perfil, adicione tags como pares de chave-valor. Para obter mais informações sobre o uso de tags emIAM, consulte IAMRecursos de marcação no Guia do IAM usuário.
13. Reveja a função e escolha Criar função.

O JSON documento a seguir é um exemplo de uma política de confiança que QLDB permite assumir uma IAM função com permissões específicas anexadas a ela.

```
{
    "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**
>
> O exemplo a seguir mostra como é possível usar as chaves de contexto de condição globais `aws:SourceArn` e `aws:SourceAccount` para evitar o problema de substituto confuso. Com essa política de confiança, QLDB pode assumir a função de qualquer QLDB fluxo na conta `myExampleLedger` somente `123456789012` para o livro contábil.

Para obter mais informações, consulte Prevenção do problema do substituto confuso entre serviços.

Depois de criar sua IAM função, retorne ao QLDB console e atualize a página Criar QLDB stream para que ele possa encontrar sua nova função.

Erros comuns em fluxos de diários em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve os erros de tempo de execução que são gerados pela Amazon QLDB para solicitações de stream de diários.

Veja a seguir uma lista de exceções comuns retornadas pelo serviço. Cada exceção inclui a mensagem de erro específica, seguida por uma breve descrição e sugestões de possíveis soluções.

AccessDeniedException

Mensagem: Usuário: *`userARN`* não está autorizado a executar: iam: PassRole no recurso: *`roleARN`*

Você não tem permissões para passar uma IAM função para o QLDB serviço. QLDBrequer uma função para todas as solicitações de stream do diário, e você deve ter permissões para transmitir essa função paraQLDB. A função QLDB fornece permissões de gravação em seu recurso específico do Amazon Kinesis Data Streams.

Verifique se você define uma IAM política que conceda permissão para realizar a `PassRole` API operação no recurso de IAM função especificado para o QLDB serviço (`qldb.amazonaws.com`). Para ver um exemplo de política, consulte Exemplos de políticas baseadas em identidade para a Amazon QLDB.

IllegalArgumentException

Mensagem: QLDB encontrou um erro ao validar o Kinesis Data Streams: Resposta do Kinesis: *`errorCode errorMessage`*

Uma possível causa desse erro é que o recurso Kinesis Data Streams fornecido não existe. Ou QLDB não tem permissões suficientes para gravar registros de dados no stream de dados especificado do Kinesis.

Verifique se o fluxo de dados do Kinesis que você fornece na sua solicitação de fluxo está correto. Para obter mais informações, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Além disso, verifique se você define uma política para o stream de dados do Kinesis especificado que conceda ao QLDB service (`qldb.amazonaws.com`) permissões para as seguintes ações. Para obter mais informações, consulte Permissões de fluxo.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Mensagem: Resposta inesperada do Kinesis Data Streams ao validar a configuração do Kinesis. Resposta do Kinesis: *`errorCode errorMessage`*

A tentativa de gravar registros de dados no fluxo de dados do Kinesis fornecido falhou com a resposta de erro fornecida do Kinesis. Para obter mais informações sobre as causas possíveis, consulte Solução de problemas dos produtores do Amazon Kinesis Data Streams no Guia do usuário do Amazon Kinesis Data Streams.

IllegalArgumentException

Mensagem: A data de início não deve ser maior que a data de término.

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de data e hora ISO8601 e no Tempo Universal Coordenado ()UTC.

IllegalArgumentException

Mensagem: A data de início não pode ser no futuro

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de `ISO 8601` data e hora e emUTC.

LimitExceededException

Mensagem: Excedeu o limite de 5 fluxos de registro em execução simultânea no Kinesis Data Streams

QLDBimpõe um limite padrão de cinco fluxos de diário simultâneos.

Gerenciamento de livros contábeis na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este capítulo descreve como usar o QLDBAPI, o AWS Command Line Interface (AWS CLI) e fazer operações de gerenciamento AWS CloudFormation de livros contábeis na AmazonQLDB.

Você também pode executar essas mesmas tarefas usando o AWS Management Console. Para ter mais informações, consulte Acessando a Amazon QLDB usando o console.

Tópicos

- Operações básicas para livros contábeis da Amazon QLDB
- Criação de QLDB recursos da Amazon com AWS CloudFormation
- Marcação de recursos da Amazon QLDB

Operações básicas para livros contábeis da Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode usar o QLDB API ou o AWS Command Line Interface (AWS CLI) para criar, atualizar e excluir livros contábeis na Amazon. QLDB Você também pode listar todos os ledgers da sua conta ou obter informações sobre um ledger específico.

Os tópicos a seguir fornecem exemplos de códigos curtos que mostram etapas comuns para operações contábeis usando o AWS SDK for Java e o. AWS CLI

Tópicos

- Criar um ledger
- Descrever um ledger
- Atualizar um ledger
- Atualizar o modo de permissões de um ledger
- Excluir um ledger
- Listar ledgers

Para exemplos de código que demonstram essas operações em um aplicativo de amostra completo, consulte os seguintes Conceitos básicos do driver tutoriais e GitHub repositórios:

- Java: Tutorial | GitHub repositório
- Node.js: Tutorial | GitHub repositório
- Python: Tutorial | repositório GitHub

Criar um ledger

Use a operação `CreateLedger` para criar um ledger no Conta da AWS. Você deve fornecer as seguintes informações:

- Nome do ledgers: o nome do ledger que você deseja criar em sua conta. O nome deve ser exclusivo entre todos os seus ledgers na Região da AWS atual.

  Restrições de nomenclatura para nomes de ledgers são definidas em Cotas e limites na Amazon QLDB.
- Modo de permissões: o modo de permissões a ser atribuído ao ledger. Escolha uma das seguintes opções:
  - Permitir tudo — Um modo de permissões antigo que permite o controle de acesso com granularidade API de nível para livros contábeis.

    Esse modo permite que os usuários que têm `SendCommand` API permissão para esse registro executem todos os comandos do PartiQL (portanto`ALLOW_ALL`,) em qualquer tabela no livro contábil especificado. Esse modo ignora qualquer política de IAM permissões em nível de tabela ou de comando que você cria para o livro contábil.

- Padrão - (Recomendado) Um modo de permissões que permite o controle do acesso com granularidade mais fina para ledgers, tabelas e comandos PartiQL. Recomendamos o uso deste modo de permissões para maximizar a segurança dos dados do seu ledger.

  Por padrão, esse modo nega todas as solicitações de execução de comandos do PartiQL em qualquer tabela nesse ledger. Para permitir comandos do PartiQL, você deve criar políticas de IAM permissões para recursos específicos da tabela e ações do PartiQL, além da `SendCommand` API permissão para o livro contábil. Para obter mais informações, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

- Proteção contra exclusão: (opcional) o sinalizador que impede que um ledger seja excluído por qualquer usuário. Se você não especificar na criação do ledger, esse atributo estará habilitado (`true`) por padrão.

  Se a proteção contra exclusão estiver ativada, você deverá desabilitá-la antes de excluir o ledger. Você pode desabilitá-la usando a operação `UpdateLedger` para definir o sinalizador como `false`.

- AWS KMS key: (opcional) a chave em AWS Key Management Service (AWS KMS) a ser usada para criptografia de dados em repouso. Escolha um dos seguintes tipos de AWS KMS keys:

  - AWS KMSchave de propriedade — Use uma KMS chave de propriedade e gerenciada AWS em seu nome.

    Se você não definir esse parâmetro durante a criação do ledger, ele usará esse tipo de chave por padrão. Você também pode usar a string `AWS_OWNED_KMS_KEY` para especificar esse tipo de chave. Essa opção não requer configuração adicional.

  - KMSChave gerenciada pelo cliente — use uma KMS chave de criptografia simétrica em sua conta que você cria, possui e gerencia. QLDBnão suporta chaves assimétricas.

    Essa opção exige que você crie uma KMS chave ou use uma chave existente na sua conta. Para obter instruções sobre como criar uma chave gerenciada pelo cliente, consulte Criação de KMS chaves de criptografia simétricas no Guia do AWS Key Management Service desenvolvedor.

    Você pode especificar uma KMS chave gerenciada pelo cliente usando um ID, alias ou Amazon Resource Name (ARN). Para saber mais, consulte Identificadores de chave (KeyId) no Guia do AWS Key Management Service desenvolvedor.

> **Note**
>
> Chaves entre regiões não são compatíveis. A KMS chave especificada deve estar na Região da AWS mesma do seu livro contábil.

Para ter mais informações, consulte Criptografia em repouso na Amazon QLDB.

- Tags: (optional) adicionar metadados à função anexando etiquetas como pares de chave-valor. Você pode adicionar tags ao seu ledger para ajudar a organizá-lo e identificá-lo. Para ter mais informações, consulte Marcação de recursos da Amazon QLDB.

O livro não está pronto para uso até que seja QLDB criado e definido seu status como`ACTIVE`.

Criar um ledger (Java)

Para criar um livro contábil usando o AWS SDK for Java

1. Crie uma instância da classe `AmazonQLDB`.
2. Crie uma instância da classe `CreateLedgerRequest` para fornecer as informações solicitadas.

   Você deve fornecer o nome do ledger e um modo de permissões.
3. Execute o método `createLedger` fornecendo o objeto de solicitação como um parâmetro.

A solicitação `createLedger` retorna um objeto `CreateLedgerResult` que tem informações sobre o ledger. Consulte a próxima seção para obter um exemplo de como usar a operação `DescribeLedger` para verificar o status do seu ledger depois de criá-lo.

O exemplo a seguir demonstra as etapas anteriores.

Example — Use as configurações padrão

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> O ledger usa as seguintes configurações padrão se você não as especificar:
>
> - Proteção contra exclusão – ativada (`true`).
> - KMSchave — KMS chave AWS de propriedade.

Example — Desative a proteção contra exclusão, use uma KMS chave gerenciada pelo cliente e anexe etiquetas

```
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);
```

Criar um ledger (AWS CLI)

Crie um novo ledger chamado `vehicle-registration` usando as configurações padrão.

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> O ledger usa as seguintes configurações padrão se você não as especificar:
>
> - Proteção contra exclusão – ativada (`true`).
> - KMSchave — KMS chave AWS de propriedade.

Ou crie um novo livro contábil chamado `vehicle-registration` com a proteção de exclusão desativada, com uma KMS chave gerenciada pelo cliente especificada e com tags especificadas.

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
```

Criar um ledger (AWS CloudFormation)

Você também pode usar um modelo AWS CloudFormation para criar ledgers. Para obter mais informações, consulte o recurso AWS::QLDB: :Ledger no Guia do AWS CloudFormation Usuário.

Descrever um ledger

Para visualizar detalhes sobre um ledger, use a operação `DescribeLedger`. Você deve fornecer o nome do ledger. A saída de `DescribeLedger` está no mesmo formato de `CreateLedger`. Isso inclui as informações a seguir:

- Nome do ledger: o nome do ledger que você deseja descrever.
- ARN— O Amazon Resource Name (ARN) para o livro contábil no seguinte formato.

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

- Proteção contra exclusão: o sinalizador que indica se o atributo de proteção contra exclusão está ativado.
- Data e hora da criação: a data e a hora, em formato epoch time, quando o ledger foi criado.
- Estado: o status atual do ledger. Pode ter um dos valores a seguir:
  - `CREATING`
  - `ACTIVE`
  - `DELETING`
  - `DELETED`

- Modo de permissões: o modo de permissões atribuído ao ledger. Pode ter um dos valores a seguir:
  - `ALLOW_ALL`— Um modo de permissões antigo que permite o controle de acesso com granularidade API de nível para livros contábeis.
  - `STANDARD`: um modo de permissões que permite o controle do acesso com granularidade mais fina para ledgers, tabelas e comandos PartiQL.
- Descrição da criptografia: informações sobre a criptografia de dados em repouso no ledger. Isso inclui os seguintes itens:
  - AWS KMS key ARN— A ARN KMS chave gerenciada pelo cliente que o livro contábil usa para criptografia em repouso. Se isso for indefinido, o livro contábil usa uma KMS chave AWS própria para criptografia.
  - Status da criptografia: o status atual da criptografia em repouso para o ledger. Pode ter um dos valores a seguir:
    - `ENABLED`: a criptografia é totalmente ativada usando a chave especificada.
    - `UPDATING`: a alteração de chave especificada está sendo processada ativamente.

      As principais alterações QLDB são assíncronas. O ledger é totalmente acessível sem nenhum impacto no desempenho enquanto a alteração da chave está sendo processada. O tempo necessário para atualizar uma chave varia dependendo do tamanho do ledger.
    - `KMS_KEY_INACCESSIBLE`— A KMS chave gerenciada pelo cliente especificada não está acessível e o livro contábil está danificado. Ou a chave foi desativada ou excluída, ou as concessões da chave foram revogadas. Quando um ledger está danificado, ele não está acessível e não aceita nenhuma solicitação de leitura ou gravação.

      Um ledger danificado retorna automaticamente ao estado ativo depois que você restaura as concessões na chave ou depois de reativar a chave que foi desativada. No entanto, a exclusão de uma KMS chave gerenciada pelo cliente é irreversível. Depois que uma chave é excluída, não é mais possível acessar os ledgers que estão protegidos com ela, e os dados ficam irrecuperáveis permanentemente.
  - Inacessível AWS KMS key — A data e a hora, no formato de época, quando a KMS chave ficou inacessível pela primeira vez, no caso de um erro.

    Isso é indefinido se a KMS chave estiver acessível.

Para ter mais informações, consulte Criptografia em repouso na Amazon QLDB.

> **Note**
>
> Depois de criar um QLDB livro contábil, ele fica pronto para uso quando seu status muda de `CREATING` para`ACTIVE`.

Descrever um ledger (Java)

Para descrever um livro contábil usando o AWS SDK for Java

1. Crie uma instância da classe `AmazonQLDB`. Ou você pode usar a mesma instância do cliente `AmazonQLDB` que você instanciou para a solicitação `CreateLedger`.
2. Crie uma instância da classe `DescribeLedgerRequest` e forneça o nome do ledger que deseja excluir.
3. Execute o método `describeLedger` fornecendo o objeto de solicitação como um parâmetro.
4. A solicitação `describeLedger` retorna um objeto `DescribeLedgerResult` que tem informações sobre o ledger.

O exemplo de código a seguir demonstra as etapas anteriores. Você pode chamar o método `describeLedger` do cliente para obter informações do ledger a qualquer momento.

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());
```

Descrever um ledger (AWS CLI)

Descrever o ledger `vehicle-registration` que você acabou de criar.

Example

```
aws qldb describe-ledger --name vehicle-registration
```

Atualizar um ledger

Atualmente, a operação `UpdateLedger` permite que você altere as seguintes configurações para um ledger existente:

- Proteção contra exclusão: o sinalizador que impede que um ledger seja excluído por qualquer usuário. Se esse atributo estiver habilitado, você deverá primeiro desabilitá-lo definindo o sinalizador como `false` antes de excluir o ledger.

  Se você não definir esse parâmetro, nenhuma alteração será feita na configuração de proteção contra exclusão do ledger.
- AWS KMS key— A chave in AWS Key Management Service (AWS KMS) a ser usada para criptografia de dados em repouso. Se você não definir esse parâmetro, nenhuma alteração será feita na KMS chave do livro contábil.

  > **Note**
  >
  > A Amazon QLDB lançou o suporte para gerenciamento de clientes AWS KMS keys em 22 de julho de 2021. Todos os livros contábeis criados antes do lançamento são protegidos Chaves pertencentes à AWS por padrão, mas atualmente não estão qualificados para criptografia em repouso usando chaves gerenciadas pelo cliente.
  > Você pode ver a hora de criação do seu livro contábil no QLDB console.

  Use uma das seguintes opções:

  - AWS KMSchave de propriedade — Use uma KMS chave de propriedade e gerenciada AWS em seu nome. Para usar esse tipo de chave, especifique a string `AWS_OWNED_KMS_KEY` desse parâmetro. Essa opção não requer configuração adicional.
  - KMSChave gerenciada pelo cliente — use uma KMS chave de criptografia simétrica em sua conta que você cria, possui e gerencia. QLDBnão suporta chaves assimétricas.

Essa opção exige que você crie uma KMS chave ou use uma chave existente na sua conta. Para obter instruções sobre como criar uma chave gerenciada pelo cliente, consulte Criação de KMS chaves de criptografia simétricas no Guia do AWS Key Management Service desenvolvedor.

Você pode especificar uma KMS chave gerenciada pelo cliente usando um ID, alias ou Amazon Resource Name (ARN). Para saber mais, consulte Identificadores de chave (KeyId) no Guia do AWS Key Management Service desenvolvedor.

> **Note**
>
> Chaves entre regiões não são compatíveis. A KMS chave especificada deve estar na Região da AWS mesma do seu livro contábil.

As principais alterações QLDB são assíncronas. O ledger é totalmente acessível sem nenhum impacto no desempenho enquanto a alteração da chave está sendo processada.

Você pode trocar de chaves tantas vezes quantas precisar, mas o tempo necessário para atualizar uma chave varia dependendo do tamanho do ledger. Você pode usar a operação `DescribeLedger` para verificar o status da criptografia em repouso.

Para ter mais informações, consulte Criptografia em repouso na Amazon QLDB.

A saída de `UpdateLedger` está no mesmo formato de `CreateLedger`.

Atualizar um ledger (Java)

Para atualizar um livro contábil usando o AWS SDK for Java

1. Crie uma instância da classe `AmazonQLDB`.
2. Crie uma instância da classe `UpdateLedgerRequest` para fornecer as informações solicitadas.

   Você deve fornecer o nome do livro contábil junto com um novo valor booleano para proteção contra exclusão ou um novo valor de cadeia de caracteres para a chave. KMS
3. Execute o método `updateLedger` fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código a seguir demonstra as etapas anteriores. A solicitação `updateLedger` retorna um objeto `UpdateLedgerResult` que tem informações atualizadas sobre o ledger.

Example – Habilitar proteção contra exclusão

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example — Use uma KMS chave gerenciada pelo cliente

```
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 — Use uma AWS KMS chave própria

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

Atualizar um ledger (AWS CLI)

Se a proteção contra exclusão de seu ledger `vehicle-registration` estiver habilitada, você deverá desabilitá-la antes de excluir o ledger.

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

Você também pode alterar as configurações de criptografia em repouso do livro contábil para usar uma KMS chave gerenciada pelo cliente.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Ou você pode alterar as configurações de criptografia em repouso para usar uma AWS KMS chave própria.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

Atualizar o modo de permissões de um ledger

A operação `UpdateLedgerPermissionsMode` permite que você altere o modo de permissões de um ledger existente. Escolha uma das seguintes opções:

- Permitir tudo — Um modo de permissões antigo que permite o controle de acesso com granularidade API de nível para livros contábeis.

  Esse modo permite que os usuários que têm `SendCommand` API permissão para esse registro executem todos os comandos do PartiQL (portanto`ALLOW_ALL`,) em qualquer tabela no livro contábil especificado. Esse modo ignora qualquer política de IAM permissões em nível de tabela ou de comando que você cria para o livro contábil.
- Padrão - (Recomendado) Um modo de permissões que permite o controle do acesso com granularidade mais fina para ledgers, tabelas e comandos PartiQL. Recomendamos o uso deste modo de permissões para maximizar a segurança dos dados do seu ledger.

  Por padrão, esse modo nega todas as solicitações de execução de comandos do PartiQL em qualquer tabela nesse ledger. Para permitir comandos do PartiQL, você deve criar políticas de IAM permissões para recursos específicos da tabela e ações do PartiQL, além da `SendCommand` API permissão para o livro contábil. Para obter mais informações, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

> **Important**
>
> Antes de mudar para o modo de `STANDARD` permissões, você deve primeiro criar todas as IAM políticas e tags de tabela necessárias para evitar interrupções para seus usuários. Para saber mais, vá para Migrar para o modo de permissões padrão.

Atualizar o modo de permissões de um ledger (Java)

Para atualizar um modo de permissões contábeis usando o AWS SDK for Java

1. Crie uma instância da classe `AmazonQLDB`.
2. Crie uma instância da classe `UpdateLedgerPermissionsModeRequest` para fornecer as informações solicitadas.

   Você deve fornecer o nome do ledger junto com um novo valor de string para o modo de permissões.
3. Execute o método `updateLedgerPermissionsMode` fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código a seguir demonstra as etapas anteriores. A solicitação `updateLedgerPermissionsMode` retorna um objeto `UpdateLedgerPermissionsModeResult` que tem informações atualizadas sobre o ledger.

Example — Atribuir o modo de permissões padrão

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

Atualizar o modo de permissões de um ledger (AWS CLI)

Atribui o modo de permissões `STANDARD` ao seu ledger `vehicle-registration`.

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

Migrar para o modo de permissões padrão

Para migrar para o modo de `STANDARD` permissões, recomendamos analisar seus padrões de QLDB acesso e adicionar IAM políticas que concedam aos usuários as permissões apropriadas para acessar seus recursos.

Antes de mudar para o modo de `STANDARD` permissões, você deve primeiro criar todas as IAM políticas e tags de tabela necessárias. Caso contrário, alternar o modo de permissões pode atrapalhar os usuários até que você crie as IAM políticas corretas ou reverta o modo de permissões. `ALLOW_ALL` Para obter informações sobre como criar essas políticas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Você também pode usar uma política AWS gerenciada para conceder acesso total a todos os QLDB recursos. As políticas `AmazonQLDBFullAccess` `AmazonQLDBConsoleFullAccess` gerenciadas incluem todas as QLDB ações, incluindo todas as ações do PartiQL. Anexar uma dessas políticas a uma entidade principal é equivalente ao modo de `ALLOW_ALL` permissões dessa entidade principal. Para ter mais informações, consulte AWS políticas gerenciadas para a Amazon QLDB.

Excluir um ledger

Use a operação `DeleteLedger` para excluir um ledger e todo o seu conteúdo. Excluir um ledger é uma operação irrecuperável.

Se a proteção contra exclusão estiver habilitada para o seu ledger, você deverá desabilitá-la antes de excluir o ledger.

Quando você emite uma solicitação `DeleteLedger`, o status do ledger muda de `ACTIVE` para `DELETING`. Pode demorar um pouco para excluir o ledger, dependendo da quantidade de armazenamento que ele usa. Quando a `DeleteLedger` operação é concluída, o livro não existe mais em. QLDB

Excluir um ledger (Java)

Para excluir um livro contábil usando o AWS SDK for Java

1. Crie uma instância da classe `AmazonQLDB`.

2. Crie uma instância da classe `DeleteLedgerRequest` e forneça o nome do ledger que deseja excluir.
3. Execute o método `deleteLedger` fornecendo o objeto de solicitação como um parâmetro.

O exemplo de código a seguir demonstra as etapas anteriores.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

Excluir um ledger (AWS CLI)

Exclua seu ledger `vehicle-registration`.

Example

```
aws qldb delete-ledger --name vehicle-registration
```

Listar ledgers

A `ListLedgers` operação retorna informações resumidas de todos os QLDB livros contábeis do atual Conta da AWS e da região.

Listar ledgers (Java)

Para listar livros contábeis em sua conta usando o AWS SDK for Java

1. Crie uma instância da classe `AmazonQLDB`.
2. Crie uma instância da classe `ListLedgersRequest`.

   Se você recebeu um valor para `NextToken` na resposta de uma chamada `ListLedgers` anterior, deverá fornecer esse valor nessa solicitação para obter a próxima página de resultados.
3. Execute o método `listLedgers` fornecendo o objeto de solicitação como um parâmetro.
4. A solicitação `listLedgers` retorna um objeto `ListLedgersResult`. Esse objeto tem uma lista de objetos `LedgerSummary` e um token de paginação que indica se há mais resultados disponíveis:

- Se `NextToken` estiver vazio, a última página de resultados foi processada e não há mais resultados.
- Se `NextToken` não estiver vazio, há mais resultados disponíveis. Para recuperar a próxima página de resultados, use o valor de `NextToken` em uma chamada `ListLedgers` subsequente.

O exemplo de código a seguir demonstra as etapas anteriores.

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);
```

Listar ledgers (AWS CLI)

Liste todos os livros contábeis no atual Conta da AWS e na região.

Example

```
aws qldb list-ledgers
```

Criação de QLDB recursos da Amazon com AWS CloudFormation

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

QLDBA Amazon está integrada com AWS CloudFormation um serviço que ajuda você a modelar e configurar seus AWS recursos para que você possa gastar menos tempo criando e gerenciando seus recursos e infraestrutura. Você cria um modelo que descreve todos os AWS recursos que você deseja (como QLDB livros contábeis) e AWS CloudFormation provisiona e configura esses recursos para você.

Ao usar AWS CloudFormation, você pode reutilizar seu modelo para configurar seus QLDB recursos de forma consistente e repetida. Descreva seus recursos uma vez e, em seguida, provisione os mesmos recursos repetidamente em várias Contas da AWS regiões.

QLDBe AWS CloudFormation modelos

Para provisionar e configurar recursos QLDB e serviços relacionados, você deve entender AWS CloudFormation os modelos. Os modelos são arquivos de texto formatados em JSON ouYAML. Esses modelos descrevem os recursos que você deseja provisionar em suas AWS CloudFormation pilhas. Se você não estiver familiarizado com JSON ouYAML, você pode usar o AWS CloudFormation Designer para ajudá-lo a começar a usar AWS CloudFormation modelos. Para obter mais informações, consulte O que é o AWS CloudFormation Designer? no Guia do usuário do AWS CloudFormation .

QLDBsuporta a criação de livros contábeis e fluxos de diários em. AWS CloudFormationPara obter mais informações, incluindo exemplos JSON e YAML modelos de livros contábeis e fluxos, consulte as seguintes referências de tipos de recursos no Guia do AWS CloudFormation usuário:

- AWS::QLDB: :Referência do Ledger
- AWS::QLDB: :Referência de stream

Saiba mais sobre AWS CloudFormation

Para saber mais sobre isso AWS CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- AWS CloudFormation Guia do usuário
- AWS CloudFormation APIReferência
- AWS CloudFormation Guia do usuário da interface de linha de comando

Marcação de recursos da Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Uma tag é um rótulo de atributo personalizado que você atribui ou AWS atribui a um AWS recurso. Cada tag tem duas partes:

- Uma chave de tag (por exemplo `CostCenter`, `Environment` ou `Project`). Chaves de tag fazem distinção entre maiúsculas e minúsculas.
- Um campo opcional conhecido como um valor de tag (por exemplo, `111122223333` ou `Production`). Omitir o valor da tag é o mesmo que usar uma string vazia. Como chaves de tag, os valores das tags diferenciam maiúsculas de minúsculas.

As tags ajudam a:

- Identifique e organize seus AWS recursos. Muitos Serviços da AWS oferecem suporte à marcação, então você pode atribuir a mesma tag a recursos de serviços diferentes para indicar que os recursos estão relacionados. Por exemplo, você pode atribuir a mesma tag a um QLDB livro contábil da Amazon que você atribui a um bucket do Amazon S3.
- Acompanhe seus AWS custos. Você ativa essas tags no AWS Billing and Cost Management painel. AWS usa as tags para categorizar seus custos e entregar um relatório mensal de alocação de custos para você. Para obter mais informações, consulte Uso de tags de alocação de custos no Guia do usuário do AWS Billing.
- Controle o acesso aos seus AWS recursos com AWS Identity and Access Management (IAM). Para obter informações, consulte Controle de acesso baseado em atributos () ABAC com QLDB este guia do desenvolvedor e Controle o acesso usando IAM tags no Guia IAM do usuário.

Para obter dicas sobre como usar tags, consulte a postagem AWS Tagging Strategies no blog AWS Answers.

As seções a seguir fornecem mais informações sobre tags para a AmazonQLDB.

Tópicos

- Recursos suportados na Amazon QLDB
- Convenções de uso e nomenclatura de tags
- Como gerenciar tags
- Atribuição de tags a recursos durante a criação

Recursos suportados na Amazon QLDB

Os seguintes recursos na Amazon QLDB oferecem suporte à marcação:

- ledger
- tabela
- fluxo de diário

Para obter informações sobre como adicionar e gerenciar tags, consulte Como gerenciar tags.

Convenções de uso e nomenclatura de tags

As seguintes convenções básicas de nomenclatura e uso se aplicam ao uso de tags com recursos da AmazonQLDB:

- Cada recurso pode ter um máximo de 50 tags.
- Em todos os recursos, cada chave de tag deve ser exclusiva e pode ter apenas um valor.
- O tamanho máximo da chave da tag é 128 caracteres Unicode em UTF -8.
- O tamanho máximo do valor da tag é 256 caracteres Unicode em UTF -8.
- Os caracteres permitidos são letras, números, espaços representáveis em UTF -8 e os seguintes caracteres: .: + = @ _/- (hífen).
- As chaves e valores das tags diferenciam maiúsculas de minúsculas. Como melhor prática, adote uma estratégia para letras maiúsculas em tags e implemente-a de forma consistente em todos os tipos de recursos. Por exemplo, decida se deseja usar `Costcenter`, `costcenter` ou `CostCenter` e use a mesma convenção para todas as tags. Evite usar tags semelhantes com tratamento do tamanho de letra inconsistente.
- O `aws:` prefixo está reservado para AWS uso. Não é possível editar nem excluir a chave ou o valor de uma tag quando ela tem uma chave de tag com o prefixo `aws:`. As tags com esse prefixo não contam para as tags por limite de recurso.

Como gerenciar tags

As tags são compostas de propriedades `Value` e `Key` em um recurso. Você pode usar o QLDB console da Amazon AWS CLI, o ou o QLDB API para adicionar, editar ou excluir os valores dessas propriedades. Também é possível gerenciar as tags usando o AWS Resource Groups Editor de tags.

Para obter informações sobre como trabalhar com tags, consulte as seguintes API operações:

- ListTagsForResourcena Amazon QLDB API Reference
- TagResourcena Amazon QLDB API Reference
- UntagResourcena Amazon QLDB API Reference

Para usar o painel de QLDB marcação (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista de ledgers, escolha o nome do ledger cujas tags você deseja gerenciar.
4. Na página de detalhes do ledger, localize o cartão Tags e escolha Gerenciar etiquetas.
5. Na página Gerenciar tags, você pode adicionar, editar ou remover qualquer tag, conforme apropriado, para seu ledger. Quando as chaves e valores da tag estiverem como você deseja, selecione Salvar.

Atribuição de tags a recursos durante a criação

Para QLDB recursos que oferecem suporte à marcação, você pode definir tags enquanto cria o recurso usando o AWS Management Console AWS CLI, o ou o. QLDB API Ao marcar os recursos no momento da criação, você elimina a necessidade de executar scripts personalizados de marcação após a criação do recurso.

Depois que um recurso é marcado, você pode controlar o acesso ao recurso com base nessas tags. Por exemplo, você pode conceder acesso total somente aos recursos da tabela que tenham uma tag específica. Para obter um exemplo JSON de política, consulteAcesso total a todas as ações com base nas tags da tabela.

> **Note**
>
> Os recursos de tabela e fluxo não herdam as tags de seu recurso de ledger raiz.

Você também pode definir tags de tabela especificando-as em uma instrução partiQL `CREATE TABLE`. Para saber mais, consulte Tabelas de marcação.

Segurança na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Segurança na nuvem em AWS é a maior prioridade. Como um AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que funciona Serviços da AWS no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte do AWS programas de conformidade. Para saber mais sobre os programas de conformidade que se aplicam à AmazonQLDB, consulte AWS Serviços no escopo do Programa de Conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS service (Serviço da AWS) que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usarQLDB. Os tópicos a seguir mostram como configurar para atender QLDB aos seus objetivos de segurança e conformidade. Você também aprende a usar outros Serviços da AWS que ajudam você a monitorar e proteger seus QLDB recursos.

Tópicos

- Proteção de dados na Amazon QLDB
- Identity and Access Management para Amazon QLDB
- Registro e monitoramento na Amazon QLDB
- Validação de conformidade para a Amazon QLDB

- Resiliência na Amazon QLDB
- Segurança da infraestrutura na Amazon QLDB

Proteção de dados na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A ferramenta AWS modelo de responsabilidade compartilhada de se aplica à proteção de dados na AmazonQLDB. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todas as Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança do Serviços da AWS que você usa. Para obter mais informações sobre privacidade de dados, consulte Privacidade de dados FAQ. Para obter informações sobre proteção de dados na Europa, consulte AWS Modelo de responsabilidade compartilhada e postagem no GDPR blog sobre o AWS Blog de segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use a autenticação multifator (MFA) com cada conta.
- UseSSL/TLSpara se comunicar com AWS recursos. Exigimos TLS 1,2 e recomendamos TLS 1,3.
- Configure API e registre as atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Trabalhando com CloudTrail trilhas no AWS CloudTrail Guia do usuário.
- Use AWS soluções de criptografia, junto com todos os controles de segurança padrão dentro Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.

- Se você precisar de FIPS 140-3 módulos criptográficos validados ao acessar AWS por meio de uma interface de linha de comando ou umaAPI, use um FIPS endpoint. Para obter mais informações sobre os FIPS endpoints disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com QLDB ou outro Serviços da AWS usando o consoleAPI, AWS CLI, ou AWS SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é altamente recomendável que você não inclua informações de credenciais no URL para validar sua solicitação para esse servidor.

> **Note**
>
> Essa orientação sobre como evitar tags ou campos de formato livre para informações confidenciais se refere aos metadados de um recurso QLDB contábil, em vez dos dados armazenados no livro contábil. Seus dados armazenados em um recurso QLDB contábil não são usados para registros de faturamento ou diagnóstico.

Tópicos

- Criptografia em repouso na Amazon QLDB
- Criptografia em trânsito na Amazon QLDB

Criptografia em repouso na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Por padrão, todos os dados armazenados na Amazon QLDB são totalmente criptografados em repouso. QLDBa criptografia em repouso fornece segurança aprimorada ao criptografar

todos os dados contábeis em repouso usando chaves de criptografia no AWS Key Management Service (AWS KMS). Essa funcionalidade ajuda a reduzir a carga operacional e a complexidade envolvidas na proteção de dados confidenciais. Com a criptografia de dados em repouso, você pode criar aplicativos confidenciais do ledger que atendem a requisitos rigorosos de conformidade e regulamentação de criptografia.

A criptografia em repouso se integra com AWS KMS para gerenciar a chave de criptografia usada para proteger seus QLDB livros contábeis. Para obter mais informações sobre AWS KMS, veja AWS Key Management Service conceitos no AWS Key Management Service Guia do desenvolvedor.

EmQLDB, você pode especificar o tipo de AWS KMS key para cada recurso contábil. Ao criar um novo livro contábil ou atualizar um livro contábil existente, você pode escolher um dos seguintes tipos de KMS chaves para proteger seus dados contábeis:

- Chave pertencente à AWS: o tipo de criptografia padrão A chave é de propriedade de QLDB (sem custo adicional).
- Chave gerenciada pelo cliente — A chave é armazenada em seu Conta da AWS e é criado, de propriedade e gerenciado por você. Você tem controle total sobre a chave (AWS KMS taxas se aplicam).

> **Note**
>
> A Amazon QLDB lançou o suporte para gerenciamento de clientes AWS KMS keys em 22 de julho de 2021. Todos os livros contábeis criados antes do lançamento são protegidos por Chaves pertencentes à AWS por padrão, mas atualmente não estão qualificados para criptografia em repouso usando chaves gerenciadas pelo cliente.
> Você pode ver a hora de criação do seu livro contábil no QLDB console.

Quando você acessa um livro contábil, QLDB descriptografa os dados de forma transparente. Você pode alternar entre Chave pertencente à AWS e a chave gerenciada pelo cliente a qualquer momento. Você não precisa alterar códigos ou aplicações para usar ou gerenciar tabelas criptografadas.

Você pode especificar uma chave de criptografia ao criar um novo livro contábil ou alterar a chave de criptografia em um livro contábil existente usando o AWS Management Console QLDBAPI, o ou o AWS Command Line Interface (AWS CLI). Para obter mais informações, consulteUsando chaves gerenciadas pelo cliente na Amazon QLDB.

> **Note**
>
> Por padrão, a Amazon habilita QLDB automaticamente a criptografia em repouso usando Chaves pertencentes à AWS sem custo adicional. No entanto, AWS KMS cobranças são aplicadas pelo uso de uma chave gerenciada pelo cliente. Para obter informações sobre preços, consulte AWS Key Management Service preços.
> QLDBa criptografia em repouso está disponível em todos Regiões da AWS onde QLDB está disponível.

Tópicos

- Criptografia em repouso: como funciona na Amazon QLDB
- Usando chaves gerenciadas pelo cliente na Amazon QLDB

Criptografia em repouso: como funciona na Amazon QLDB

QLDBa criptografia em repouso criptografa seus dados usando o Advanced Encryption Standard de 256 bits (AES-256). Isso ajuda a proteger seus dados contra o acesso não autorizado ao armazenamento subjacente. Todos os dados armazenados nos QLDB livros contábeis são criptografados em repouso por padrão. A criptografia do lado do servidor é transparente, o que significa que não são necessárias alterações nos aplicativos.

A criptografia em repouso se integra com AWS Key Management Service (AWS KMS) para gerenciar a chave de criptografia usada para proteger seus QLDB livros contábeis. Ao criar um novo livro contábil ou atualizar um livro contábil existente, você pode escolher um dos seguintes tipos de AWS KMS chaves:

- Chave pertencente à AWS: o tipo de criptografia padrão A chave é de propriedade de QLDB (sem custo adicional).
- Chave gerenciada pelo cliente — A chave é armazenada em seu Conta da AWS e é criado, de propriedade e gerenciado por você. Você tem controle total sobre a chave (AWS KMS taxas se aplicam).

Tópicos

- Chave pertencente à AWS
- Chave gerenciada pelo cliente

- Como a Amazon QLDB usa subsídios em AWS KMS
- Restaurando subsídios em AWS KMS
- Considerações sobre criptografia em repouso

Chave pertencente à AWS

Chaves pertencentes à AWS não estão armazenados em seu Conta da AWS. Eles fazem parte de uma coleção de KMS chaves que AWS possui e gerencia para uso em vários Contas da AWS. Serviços da AWS pode usar Chaves pertencentes à AWS para proteger seus dados.

Você não precisa criar ou gerenciar Chaves pertencentes à AWS. No entanto, você não pode visualizar ou rastrear Chaves pertencentes à AWS, ou audite seu uso. Não é cobrada uma taxa mensal ou uma taxa de uso para Chaves pertencentes à AWS, e eles não contam contra o AWS KMS cotas para sua conta.

Para ter mais informações, consulte Chaves pertencentes à AWS no AWS Key Management Service Guia do desenvolvedor.

Chave gerenciada pelo cliente

As chaves gerenciadas pelo cliente são KMS chaves em seu Conta da AWS que você cria, possui e gerencia. Você tem controle total sobre essas KMS chaves. QLDBsuporta somente KMS chaves de criptografia simétricas.

Use uma chave gerenciada pelo cliente para obter os seguintes recursos:

- Definir e manter as principais IAM políticas, políticas e concessões para controlar o acesso à chave
- Habilitar e desabilitar a chave
- Material criptográfico rotativo para a chave
- Criação de tags-chave e aliases
- Agendar a exclusão da chave
- Importar seu próprio material de chave ou usar um armazenamento de chaves personalizado de sua propriedade e que você gerencia.
- O uso do AWS CloudTrail e Amazon CloudWatch Logs para rastrear as solicitações QLDB enviadas para AWS KMS em seu nome

Para obter mais informações, consulte Chaves gerenciadas pelo cliente no AWS Key Management Service Guia do desenvolvedor.

As chaves gerenciadas pelo cliente incorrem em uma cobrança por cada API chamada, e AWS KMS as cotas se aplicam a essas KMS chaves. Para ter mais informações, consulte AWS KMS cotas de recursos ou solicitações.

Quando você especifica uma chave gerenciada pelo cliente como a KMS chave para um livro contábil, todos os dados contábeis no armazenamento em diário e no armazenamento indexado são protegidos com a mesma chave gerenciada pelo cliente.

Chaves gerenciadas pelo cliente inacessíveis

Se você desativar sua chave gerenciada pelo cliente, programar a chave para exclusão ou revogar as concessões da chave, o status da criptografia contábil será `KMS_KEY_INACCESSIBLE`. Nesse estado, o ledger está comprometido e não aceita nenhuma solicitação de leitura ou gravação. Uma chave inacessível impede que todos os usuários e o QLDB serviço criptografem ou descriptografem dados e realizem operações de leitura e gravação no livro contábil. QLDBdeve ter acesso à sua KMS chave para garantir que você possa continuar acessando seu livro contábil e evitar a perda de dados.

> **Important**
>
> Um ledger danificado retorna automaticamente ao estado ativo depois que você restaura as concessões na chave ou depois de reativar a chave que foi desativada.
> No entanto, a exclusão de uma chave gerenciada pelo cliente é irreversível. Depois que uma chave é excluída, não é mais possível acessar os ledgers que estão protegidos com ela, e os dados ficam irrecuperáveis permanentemente.

Para verificar o status da criptografia de um livro contábil, use o AWS Management Console ou a DescribeLedgerAPIoperação.

Como a Amazon QLDB usa subsídios em AWS KMS

QLDBexige subsídios para usar sua chave gerenciada pelo cliente. Quando você cria um livro contábil protegido com uma chave gerenciada pelo cliente, QLDB cria concessões em seu nome enviando CreateGrantsolicitações para AWS KMS. Subsídios em AWS KMS são usados para dar QLDB acesso a uma KMS chave em um cliente Conta da AWS. Para obter mais informações, consulte Usando subsídios no AWS Key Management Service Guia do desenvolvedor.

QLDBexige que as concessões usem sua chave gerenciada pelo cliente para o seguinte AWS KMS operações:

- DescribeKey— Verifique se a KMS chave de criptografia simétrica especificada é válida.
- GenerateDataKey— Gere uma chave de dados simétrica exclusiva QLDB usada para criptografar dados em repouso em seu livro contábil.
- Descriptografar: descriptografa dados que foram criptografados com a chave gerenciada pelo cliente.
- Criptografar: criptografe texto simples em texto cifrado usando sua chave gerenciada pelo cliente.

É possível revogar o acesso à concessão, ou remover o acesso do serviço à chave gerenciada pelo cliente quando você quiser. Se você fizer isso, a chave ficará inacessível e QLDB perderá o acesso a qualquer um dos dados contábeis protegidos pela chave gerenciada pelo cliente. Nesse estado, o ledger está comprometido e não aceita nenhuma solicitação de leitura ou gravação até que você restaure as concessões na chave.

Restaurando subsídios em AWS KMS

Para restaurar concessões em uma chave gerenciada pelo cliente e recuperar o acesso a um livro contábil emQLDB, você pode atualizar o livro contábil e especificar a mesma KMS chave. Para obter instruções, consulte Atualizando o AWS KMS key de um livro contábil existente.

Considerações sobre criptografia em repouso

Considere o seguinte ao usar criptografia em repouso emQLDB:

- A criptografia do lado do servidor em repouso é ativada por padrão em todos os dados QLDB contábeis e não pode ser desativada. Não é possível criptografar apenas um subconjunto de itens em um ledger.
- A criptografia em repouso só criptografa dados enquanto eles estão estáticos (em repouso) em uma mídia de armazenamento persistente. Se a segurança dos dados for motivo de preocupação para dados em trânsito ou dados em uso, talvez seja necessário tomar outras medidas, como:
  - Dados em trânsito: todos os seus dados QLDB são criptografados em trânsito. Por padrão, as comunicações de e para QLDB usam o HTTPS protocolo, que protege o tráfego da rede usando a criptografia Secure Sockets Layer (SSL) /Transport Layer Security (TLS).
  - Dados em uso: proteja seus dados antes de enviá-los QLDB usando criptografia do lado do cliente.

Para saber como implementar chaves gerenciadas pelo cliente para ledgers, vá para Usando chaves gerenciadas pelo cliente na Amazon QLDB.

Usando chaves gerenciadas pelo cliente na Amazon QLDB

Você pode usar o AWS Management Console, o AWS Command Line Interface (AWS CLI), ou QLDB API para especificar o AWS KMS key para novos livros contábeis e livros contábeis existentes na Amazon. QLDB Os tópicos a seguir descrevem como gerenciar e monitorar o uso das chaves gerenciadas pelo cliente emQLDB.

Tópicos

- Pré-requisitos
- Especificando o AWS KMS key para um novo livro
- Atualizando o AWS KMS key de um livro contábil existente
- Monitorando seu AWS KMS keys

Pré-requisitos

Antes de proteger um QLDB livro contábil com uma chave gerenciada pelo cliente, você deve primeiro criar a chave no AWS Key Management Service (AWS KMS). Você também deve especificar uma política chave que permita QLDB criar concessões com base nisso. AWS KMS key em seu nome.

Criar uma chave gerenciada pelo cliente

Para criar uma chave gerenciada pelo cliente, siga as etapas em Criação de KMS chaves de criptografia simétricas no AWS Key Management Service Guia do desenvolvedor. QLDBnão suporta chaves assimétricas.

Definir uma política de chaves

As principais políticas são a principal forma de controlar o acesso às chaves gerenciadas pelo cliente no AWS KMS. Cada chave gerenciada pelo cliente deve ter exatamente uma política de chaves. As declarações no documento de política principal determinam quem tem permissão para usar a KMS chave e como eles podem usá-la. Para obter mais informações, consulte Usando políticas de chaves em AWS KMS.

Você pode especificar uma política de chaves quando criar uma chave gerenciada pelo cliente. Para alterar uma política de chave para uma chave gerenciada pelo cliente existente, consulte Alterar uma política de chave.

QLDBPara permitir o uso da chave gerenciada pelo cliente, a política de chaves deve incluir permissões para o seguinte: AWS KMS ações:

- kms: CreateGrant — Adiciona uma concessão a uma chave gerenciada pelo cliente. Concede acesso de controle a uma KMS chave especificada.

  Quando você cria ou atualiza um livro contábil com uma chave gerenciada pelo cliente especificada, QLDB cria concessões que permitem acesso às operações de concessão necessárias. As operações de concessão incluem o seguinte:

  - GenerateDataKey
  - Decrypt
  - Encrypt
- kms: DescribeKey — Retorna informações detalhadas sobre uma chave gerenciada pelo cliente. QLDBusa essas informações para validar a chave.

Exemplo de política de chave

Veja a seguir um exemplo de política importante que você pode usar paraQLDB. Essa política permite que diretores autorizados a usar QLDB a conta liguem `111122223333` para `DescribeKey` e `CreateGrant` operem no recurso`arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`.

Para usar essa política, substitua *`us-east-1`*, *`111122223333`* e *`1234abcd-12ab-34cd-56ef-1234567890ab`* no exemplo com suas próprias informações.

```
{
    "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"
                }
            }
        }
    ]
}
```

Especificando o AWS KMS key para um novo livro

Siga estas etapas para especificar uma KMS chave ao criar um novo livro contábil usando o QLDB console ou o AWS CLI.

Você pode especificar uma chave gerenciada pelo cliente usando um ID, alias ou Amazon Resource Name (ARN). Para saber mais, consulte Identificadores de chave (KeyId) no AWS Key Management Service Guia do desenvolvedor.

> **Note**
>
> Chaves entre regiões não são compatíveis. A KMS chave especificada deve estar na mesma Região da AWS como seu livro contábil.

Criar um ledger (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. Escolha (Criar ledger).
3. Na página Criar Ledger, faça o seguinte:
   - Informações do livro contábil — insira um nome de livro contábil que seja exclusivo entre todos os livros contábeis do atual. Conta da AWS e região.
   - Modo de permissões: escolha um modo de permissões para atribuir ao ledger:

    - Permitir tudo
    - Padrão (recomendado)
- Criptografar dados em repouso — Escolha o tipo de KMS chave a ser usada para criptografia em repouso:
    - Use AWS KMSchave de propriedade — Use uma KMS chave que seja de propriedade e gerenciada por AWS em seu nome. Essa é a opção padrão e não requer configuração adicional.
    - Escolha um diferente AWS KMS chave — Use uma KMS chave de criptografia simétrica em sua conta que você cria, possui e gerencia.

      Para criar uma nova chave usando o AWS KMS console, escolha Criar um AWS KMS chave. Para obter mais informações, consulte Criação de KMS chaves de criptografia simétricas no AWS Key Management Service Guia do desenvolvedor.

      Para usar uma KMS chave existente, escolha uma na lista suspensa ou especifique uma KMS chave. ARN

4. Quando estiver satisfeito com as configurações, escolha Criar ledger.

   Você pode acessar seu QLDB livro contábil quando seu status se tornar Ativo. Isso pode demorar vários minutos.

Criando um livro contábil (AWS CLI)

Use o comando AWS CLI para criar um livro contábil QLDB com o padrão Chave pertencente à AWS ou uma chave gerenciada pelo cliente.

Example — Para criar um livro contábil com o padrão Chave pertencente à AWS

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

Example — Para criar um ledger com uma chave gerenciada pelo cliente

```
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
```

Atualizando o AWS KMS key de um livro contábil existente

Você também pode usar o QLDB console ou o AWS CLI para atualizar a KMS chave de um livro contábil existente em um Chave pertencente à AWS ou uma chave gerenciada pelo cliente a qualquer momento.

> **Note**
>
> A Amazon QLDB lançou o suporte para gerenciamento de clientes AWS KMS keys em 22 de julho de 2021. Todos os livros contábeis criados antes do lançamento são protegidos por Chaves pertencentes à AWS por padrão, mas atualmente não estão qualificados para criptografia em repouso usando chaves gerenciadas pelo cliente.
> Você pode ver a hora de criação do seu livro contábil no QLDB console.

As principais alterações QLDB são assíncronas. O ledger é totalmente acessível sem nenhum impacto no desempenho enquanto a alteração da chave está sendo processada. O tempo necessário para atualizar uma chave varia dependendo do tamanho do ledger.

Você pode especificar uma chave gerenciada pelo cliente usando um ID, alias ou Amazon Resource Name (ARN). Para saber mais, consulte Identificadores de chave (KeyId) no AWS Key Management Service Guia do desenvolvedor.

> **Note**
>
> Chaves entre regiões não são compatíveis. A KMS chave especificada deve estar na mesma Região da AWS como seu livro contábil.

Atualizar um ledger (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista de ledgers, selecione o ledger que você deseja atualizar e escolha Editar ledger.
4. Na página Editar livro contábil, escolha o tipo de KMS chave a ser usada para criptografia em repouso:

- Use AWS KMSchave de propriedade — Use uma KMS chave que seja de propriedade e gerenciada por AWS em seu nome. Essa é a opção padrão e não requer configuração adicional.
- Escolha um diferente AWS KMS chave — Use uma KMS chave de criptografia simétrica em sua conta que você cria, possui e gerencia.

  Para criar uma nova chave usando o AWS KMS console, escolha Criar um AWS KMS chave. Para obter mais informações, consulte Criação de KMS chaves de criptografia simétricas no AWS Key Management Service Guia do desenvolvedor.

  Para usar uma KMS chave existente, escolha uma na lista suspensa ou especifique uma KMS chave. ARN

5. Escolha Confirmar alterações.

Atualizando um livro contábil (AWS CLI)

Use o comando AWS CLI para atualizar um livro contábil existente QLDB com o padrão Chave pertencente à AWS ou uma chave gerenciada pelo cliente.

Example — Para atualizar um livro contábil com o padrão Chave pertencente à AWS

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — Para atualizar um ledger com uma chave gerenciada pelo cliente

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Monitorando seu AWS KMS keys

Se você usa uma chave gerenciada pelo cliente para proteger seus QLDB livros contábeis da Amazon, você pode usar AWS CloudTrailou Amazon CloudWatch Logs para rastrear as solicitações QLDB enviadas para AWS KMS em seu nome. Para obter mais informações, consulte Monitoramento AWS KMS keys no AWS Key Management Service Guia do desenvolvedor.

Os exemplos a seguir são entradas de CloudTrail registro para as operações `CreateGrant` `GenerateDataKeyDecrypt`,`Encrypt`,, `DescribeKey` e.

CreateGrant

Quando você especifica uma chave gerenciada pelo cliente para proteger seu livro contábil, QLDB envia `CreateGrant` solicitações para AWS KMS em seu nome para permitir o acesso à sua KMS chave. Além disso, QLDB usa a `RetireGrant` operação para remover concessões quando você exclui um livro contábil.

As doações QLDB criadas são específicas de um livro contábil. A entidade principal na solicitação `CreateGrant` é o usuário que criou a tabela.

O evento que registra a operação `CreateGrant` é semelhante ao evento de exemplo a seguir. Os parâmetros incluem o Amazon Resource Name (ARN) da chave gerenciada pelo cliente, o principal beneficiário e o principal aposentado (o QLDB serviço) e as operações que a concessão cobre.

```
{
    "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

Quando você especifica uma chave gerenciada pelo cliente para proteger seu livro contábil, QLDB cria uma chave de dados exclusiva. Ele envia uma `GenerateDataKey` solicitação para AWS KMS que especifica a chave gerenciada pelo cliente para o livro contábil.

O evento que registra a operação `GenerateDataKey` é semelhante ao evento de exemplo a seguir. O usuário é a conta QLDB de serviço. Os parâmetros incluem a chave gerenciada pelo

cliente, um especificador ARN de chave de dados que exige um comprimento de 32 bytes e o contexto de criptografia que identifica o nó interno da hierarquia de chaves.


```
{
    "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

Quando você acessa um livro contábil, QLDB chama a `Decrypt` operação para descriptografar a chave de dados armazenada do livro contábil para que ele possa acessar os dados criptografados no livro contábil.

O evento que registra a operação `Decrypt` é semelhante ao evento de exemplo a seguir. O usuário é a conta QLDB de serviço. Os parâmetros incluem a chave gerenciada pelo cliente e o contexto ARN de criptografia que identifica o nó interno da hierarquia de chaves.

```
{
    "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

QLDBchama a `Encrypt` operação para criptografar texto simples em texto cifrado usando sua chave gerenciada pelo cliente.

O evento que registra a operação `Encrypt` é semelhante ao evento de exemplo a seguir. O usuário é a conta QLDB de serviço. Os parâmetros incluem a ARN chave gerenciada pelo cliente e o contexto de criptografia que especifica a ID exclusiva interna do livro contábil.

```
{
    "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

QLDBchama a `DescribeKey` operação para determinar se a KMS chave que você especificou existe no Conta da AWS e região.

O evento que registra a operação `DescribeKey` é semelhante ao evento de exemplo a seguir. O principal é o usuário em seu Conta da AWS quem especificou a KMS chave. Os parâmetros incluem a ARN chave gerenciada pelo cliente.

```
{
    "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"
}
```

Criptografia em trânsito na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB só aceita conexões seguras que usam o HTTPS protocolo, que protege o tráfego de rede usando Secure Sockets Layer (SSL) /Transport Layer Security (TLS). A criptografia em

trânsito fornece uma camada adicional de proteção de dados ao criptografar seus dados à medida que eles viajam de ida QLDB e volta. Políticas organizacionais, regulamentações setoriais ou governamentais e exigências de conformidade geralmente demandam o uso de criptografia em trânsito para aumentar a segurança de dados de seus aplicativos quando transmitem dados pela rede.

QLDBtambém oferece FIPS endpoints em regiões selecionadas. Ao contrário do padrão AWS endpoints, os FIPS endpoints usam uma biblioteca de TLS software que está em conformidade com o Federal Information Processing Standard () FIPS 140-2. Esses endpoints podem ser necessários por empresas que interagem com o governo dos Estados Unidos. Para obter mais informações, consulte FIPSendpoints no Referência geral da AWS. Para obter uma lista completa das regiões e endpoints disponíveisQLDB, consulte os QLDBendpoints e cotas da Amazon.

Identity and Access Management para Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

AWS Identity and Access Management (IAM) é um AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso ao AWS recursos. IAMos administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar QLDB os recursos. IAMé um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticando com identidades
- Gerenciando acesso usando políticas
- Como a Amazon QLDB trabalha com IAM
- Introdução ao modo de permissões padrão na Amazon QLDB
- Exemplos de políticas baseadas em identidade para a Amazon QLDB

- Prevenção do problema do substituto confuso entre serviços
- AWS políticas gerenciadas para a Amazon QLDB
- Solução de problemas de QLDB identidade e acesso da Amazon

Público

Como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você realizaQLDB.

Usuário do serviço — Se você usar o QLDB serviço para fazer seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais QLDB recursos para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um recurso noQLDB, consulteSolução de problemas de QLDB identidade e acesso da Amazon.

Administrador de serviços — Se você é responsável pelos QLDB recursos da sua empresa, provavelmente tem acesso total QLDB a. É seu trabalho determinar quais QLDB recursos e recursos seus usuários do serviço devem acessar. Em seguida, você deve enviar solicitações ao IAM administrador para alterar as permissões dos usuários do serviço. Revise as informações nesta página para entender os conceitos básicos doIAM. Para saber mais sobre como sua empresa pode usar IAM comQLDB, consulteComo a Amazon QLDB trabalha com IAM.

IAMadministrador — Se você for IAM administrador, talvez queira saber detalhes sobre como criar políticas para gerenciar o acessoQLDB. Para ver exemplos de políticas QLDB baseadas em identidade que você pode usar emIAM, consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Autenticando com identidades

Autenticação é como você faz login em AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado em AWS) como o Usuário raiz da conta da AWS, como IAM usuário ou assumindo uma IAM função.

Você pode fazer login em AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Os usuários (do IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você entra como uma identidade federada, seu

administrador configurou previamente a federação de identidades usando IAM funções. Quando você acessa AWS ao usar a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou o AWS portal de acesso. Para obter mais informações sobre como fazer login no AWS, veja Como fazer login no seu Conta da AWS no Início de Sessão da AWS Guia do usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para você mesmo assinar solicitações, consulte Assinatura AWS APIsolicitações no Guia do IAM usuário.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no AWS IAM Identity Center Guia do usuário e uso da autenticação multifatorial (MFA) em AWS no IAM Guia do usuário.

Conta da AWS usuário raiz

Quando você cria um Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS e recursos na conta. Essa identidade é chamada de Conta da AWS usuário root e é acessado fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para ver a lista completa de tarefas que exigem que você faça login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do IAM usuário.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, um provedor de identidade da web, o AWS Directory Service, o diretório do Identity Center ou qualquer usuário que acesse Serviços da AWS usando credenciais fornecidas por meio de uma fonte de identidade. Quando as identidades federadas acessam Contas da AWS, eles assumem funções, e as funções fornecem credenciais temporárias.

Para gerenciamento de acesso centralizado, recomendamos que você use AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todas as suas Contas da AWS e aplicativos. Para obter informações sobre o IAM Identity Center, consulte O que é o IAM Identity Center? no AWS IAM Identity Center Guia do usuário.

Grupos e usuários do IAM

Um IAMusuário é uma identidade dentro do seu Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos confiar em credenciais temporárias em vez de criar IAM usuários que tenham credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com IAM os usuários, recomendamos que você alterne as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exigem credenciais de longo prazo no Guia do IAMusuário.

Um IAMgrupo é uma identidade que especifica uma coleção de IAM usuários. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar IAM recursos.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um IAM usuário (em vez de uma função) no Guia do IAM usuário.

IAMfunções

Um IAMpapel é uma identidade dentro de você Conta da AWS que tem permissões específicas. É semelhante a um IAM usuário, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma IAM função no AWS Management Console trocando de papéis. Você pode assumir uma função chamando um AWS CLI ou AWS APIoperação ou usando um personalizadoURL. Para obter mais informações sobre métodos de uso de funções, consulte Usando IAM funções no Guia IAM do usuário.

IAMfunções com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa

identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter informações sobre funções para federação, consulte Criação de uma função para um provedor de identidade terceirizado no Guia IAM do usuário. Se você usa o IAM Identity Center, configura um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a uma função em. IAM Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no AWS IAM Identity Center Guia do usuário.

- Permissões temporárias IAM de IAM usuário — Um usuário ou função pode assumir uma IAM função para assumir temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas — Você pode usar uma IAM função para permitir que alguém (um diretor confiável) em uma conta diferente acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.
- Acesso entre serviços — Alguns Serviços da AWS use recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um IAM usuário ou função para realizar ações no AWS, você é considerado diretor. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinado com a solicitação AWS service (Serviço da AWS) para fazer solicitações para serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou recursos a serem concluídos. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.
  - Função de serviço — Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.
  - Função vinculada a serviços — Uma função vinculada a serviços é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode assumir a função de realizar

uma ação em seu nome. As funções vinculadas ao serviço aparecem em seu Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.

- Aplicativos em execução na Amazon EC2 — Você pode usar uma IAM função para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo AWS CLI ou AWS APIsolicitações. Isso é preferível ao armazenamento de chaves de acesso na EC2 instância. Para atribuir um AWS Ao atribuir a uma EC2 instância e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância que é anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Como usar uma IAM função para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia IAM do usuário.

Para saber se usar IAM funções ou IAM usuários, consulte Quando criar uma IAM função (em vez de um usuário) no Guia do IAM usuário.

Gerenciando acesso usando políticas

Você controla o acesso em AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto em AWS que, quando associados a uma identidade ou recurso, definem suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada em AWS como JSON documentos. Para obter mais informações sobre a estrutura e o conteúdo dos documentos de JSON política, consulte Visão geral das JSON políticas no Guia IAM do usuário.

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

IAMas políticas definem permissões para uma ação, independentemente do método usado para realizar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função do AWS Management Console, o AWS CLI, ou o AWS API.

Políticas baseadas em identidade

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas gerenciadas incluem AWS políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolha entre políticas gerenciadas e políticas em linha no Guia do IAMusuário.

Políticas baseadas no recurso

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são as políticas de confiança de IAM funções e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou Serviços da AWS.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar AWS políticas gerenciadas a partir IAM de uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Amazon S3, AWS WAF, e a Amazon VPC são exemplos de serviços que oferecem suporteACLs. Para saber maisACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões — Um limite de permissões é um recurso avançado no qual você define as permissões máximas que uma política baseada em identidade pode conceder a uma IAM entidade (IAMusuário ou função). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para IAM entidades no Guia IAM do usuário.
- Políticas de controle de serviço (SCPs) — SCPs são JSON políticas que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. Os SCP limites de permissões para entidades em contas de membros, incluindo cada Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations eSCPs, consulte Políticas de controle de serviços no AWS Organizations Guia do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia IAM do usuário.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determina se uma solicitação deve ser permitida quando vários tipos de política estão envolvidos, consulte Lógica de avaliação de políticas no Guia IAM do usuário.

Como a Amazon QLDB trabalha com IAM

Antes de usar IAM para gerenciar o acesso aoQLDB, saiba quais IAM recursos estão disponíveis para usoQLDB.

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

IAMrecursos que você pode usar com a Amazon QLDB

IAMrecurso	QLDBapoio
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de políticas	Sim
ACLs	Não
ABAC(tags nas políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Não
Perfis de serviço	Sim
Perfis vinculados ao serviço	Não

Para obter uma visão de alto nível de como QLDB e outras Serviços da AWS funciona com a maioria dos IAM recursos, consulte Serviços da AWS que funcionam com IAM o Guia IAM do Usuário.

Políticas baseadas em identidade para o QLDB

Compatível com políticas baseadas em identidade: Sim

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criação de IAM políticas no Guia do IAMusuário.

Com políticas IAM baseadas em identidade, você pode especificar ações e recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que você pode usar em uma JSON política, consulte a referência IAM JSON de elementos de política no Guia IAM do usuário.

Exemplos de políticas baseadas em identidade para o QLDB

Para ver exemplos de políticas QLDB baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Políticas baseadas em recursos no QLDB

Suporte a políticas baseadas em recursos: não

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são as políticas de confiança de IAM funções e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou Serviços da AWS.

Para habilitar o acesso entre contas, você pode especificar uma conta ou IAM entidades inteiras em outra conta como principal em uma política baseada em recursos. Adicionar uma entidade principal

entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso estão em condições diferentes Contas da AWS, um IAM administrador na conta confiável também deve conceder permissão à entidade principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Ações de políticas para o QLDB

Compatível com ações de políticas: Sim

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O `Action` elemento de uma JSON política descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome que as associadas AWS APIoperação. Há algumas exceções, como ações somente de permissão que não têm uma operação correspondente. API Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de QLDB ações, consulte Ações definidas pela Amazon QLDB na Referência de autorização de serviço.

As ações de política QLDB usam o seguinte prefixo antes da ação:

```
qldb
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "qldb:Describe*"
```

Para interagir com os dados QLDB transacionais API (QLDBSessão) executando instruções partiQL em um livro contábil, você deve conceder permissão para a ação da seguinte maneira. `SendCommand`

```
"Action": "qldb:SendCommand"
```

Para ledgers no modo de permissões `STANDARD`, consulte as permissões adicionais necessárias Referência de permissões PartiQL para cada comando partiQL.

Para ver exemplos de políticas QLDB baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Recursos de políticas para QLDB

Compatível com recursos de políticas: Sim

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` JSON de política especifica o objeto ou objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu Amazon Resource Name (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de QLDB recursos e seusARNs, consulte Recursos definidos pela Amazon QLDB na Referência de autorização de serviço. Para saber com quais ações você pode especificar cada recurso, consulte Ações definidas pela Amazon QLDB. ARN

EmQLDB, os recursos primários são livros contábeis. QLDBtambém oferece suporte a tipos de recursos adicionais: tabelas e fluxos. No entanto, você pode criar tabelas e fluxos somente no contexto de um ledger existente.

Uma QLDB tabela é uma visão materializada de uma coleção não ordenada de revisões de documentos do diário do livro contábil. No modo de `STANDARD` permissões de um livro contábil, você deve criar IAM políticas que concedam permissões para executar instruções partiQL nesse recurso de tabela. Com permissões em um recurso de tabela, você pode executar instruções que acessam o estado atual da tabela. Você também pode consultar o histórico de revisões da tabela usando a função `history()` incorporada. Para saber mais, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

> **Note**
>
> A instrução `CREATE TABLE` cria uma tabela com um ID exclusivo e o nome da tabela fornecido. O nome da tabela fornecido deve ser exclusivo entre todas as tabelas ativas. No entanto, QLDB permite que você desative tabelas, então pode haver várias tabelas inativas que compartilham o mesmo nome de tabela. Portanto, os recursos de tabela ARNs se referem ao ID exclusivo atribuído pelo sistema em vez do nome da tabela definido pelo usuário.

Cada ledger também fornece um recurso de catálogo definido pelo sistema que você pode consultar para listar todas as tabelas e índices em um ledger. Para obter mais informações sobre o modelo QLDB de objeto de dados, consulteConceitos e terminologia fundamentais na Amazon QLDB.

Esses recursos são exclusivos ARNs associados a eles, conforme mostrado na tabela a seguir.

Tipo de recurso	ARN
ledger	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}`
table	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}`

Tipo de recurso	ARN
catalog	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables`
stream	`arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}`

Por exemplo, para especificar o `myExampleLedger` recurso em sua declaração, use o seguinteARN.

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

Para especificar vários recursos em uma única instrução, separe-os ARNs com vírgulas.

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

Para ver exemplos de políticas QLDB baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Chaves de condição de políticas para QLDB

Compatível com chaves de condição de política específicas de serviço: Sim

Os administradores podem usar AWS JSONpolíticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários `Condition` elementos em uma instrução ou várias chaves em um único `Condition` elemento, AWS os avalia usando uma `AND` operação lógica. Se você especificar vários

valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, você pode conceder permissão a um IAM usuário para acessar um recurso somente se ele estiver marcado com o nome de IAM usuário. Para obter mais informações, consulte elementos de IAM política: variáveis e tags no Guia IAM do usuário.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver tudo AWS chaves de condição globais, consulte AWS chaves de contexto de condição global no Guia IAM do usuário.

Para ver uma lista de chaves de QLDB condição, consulte Chaves de condição da Amazon QLDB na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas pela Amazon QLDB.

As ações `PartiQLDropIndex` e `PartiQLDropTable` oferecem suporte à chave de condição `qldb:Purge`. Essa condição filtra o acesso pelo valor de `purge` especificado em uma instrução do PartiQL `DROP`. No entanto, QLDB atualmente suporta apenas `purge = true` `DROP INDEX` declarações e `purge = false` `DROP TABLE` declarações. Outras QLDB ações oferecem suporte a algumas chaves de condição globais.

Para ver exemplos de políticas QLDB baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Listas de controle de acesso (ACLs) em QLDB

SuportesACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Controle de acesso baseado em atributos () ABAC com QLDB

Suportes ABAC (tags nas políticas): Sim

O controle de acesso baseado em atributos (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a IAM entidades (usuários ou funções) e a muitas AWS recursos. Marcar entidades e

recursos é a primeira etapa doABAC. Em seguida, você cria ABAC políticas para permitir operações quando a tag do diretor corresponde à tag do recurso que ele está tentando acessar.

ABACé útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna complicado.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobreABAC, consulte O que éABAC? no Guia do IAM usuário. Para ver um tutorial com etapas de configuraçãoABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do IAMusuário.

Para obter mais informações sobre a marcação de QLDB recursos, consulteMarcação de recursos da Amazon QLDB.

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Atualizando QLDB livros contábeis com base em etiquetas.

Usar credenciais temporárias com QLDB

Compatível com credenciais temporárias: Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS trabalhar com credenciais temporárias, consulte Serviços da AWS que funcionam com IAM o Guia IAM do Usuário.

Você está usando credenciais temporárias se fizer login no AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre a troca de funções, consulte Alternando para uma função (console) no Guia IAM do usuário.

Você pode criar manualmente credenciais temporárias usando o AWS CLI ou AWS API. Você pode então usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias emIAM.

Permissões de entidade principal entre serviços para o QLDB

Suporta sessões de acesso direto (FAS): Não

Quando você usa um IAM usuário ou uma função para realizar ações no AWS, você é considerado diretor. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinado com a solicitação AWS service (Serviço da AWS) para fazer solicitações para serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou recursos a serem concluídos. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.

Funções de serviço para QLDB

Compatível com perfis de serviço: Sim

Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS service (Serviço da AWS) no IAM Guia do usuário.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper QLDB a funcionalidade. Edite as funções de serviço somente quando QLDB fornecer orientação para fazer isso.

QLDBsuporta funções de serviço para as `StreamJournalToKinesis` API operações `ExportJournalToS3` e, conforme descrito na seção a seguir.

Escolhendo uma IAM função em QLDB

Ao exportar ou transmitir blocos de diárioQLDB, você deve escolher uma função que permita QLDB gravar objetos em um determinado destino em seu nome. Se você já criou uma função de serviço,

QLDB fornece uma lista de funções para escolher. É importante escolher uma função que permita o acesso para gravar em seu bucket específico do Amazon S3 para uma exportação ou em seu recurso específico do Amazon Kinesis Data Streams para um fluxo. Para obter mais informações, consulte Permissões de exportação de diário em QLDB ou Permissões de transmissão em QLDB.

> **Note**
>
> Para passar uma função QLDB ao solicitar uma exportação ou transmissão de diário, você deve ter permissões para realizar a `iam:PassRole` ação no recurso da IAM função. Isso é um acréscimo às permissões para atuar `qldb:ExportJournalToS3` no recurso QLDB contábil ou `qldb:StreamJournalToKinesis` no sub-recurso de QLDB fluxo.

Funções vinculadas ao serviço para o QLDB

Compatível com perfis vinculados ao serviço: Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode assumir a função de realizar uma ação em seu nome. As funções vinculadas ao serviço aparecem em seu Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.

Para obter detalhes sobre a criação ou o gerenciamento de funções vinculadas ao serviço, consulte Serviços da AWS que funcionam com IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Introdução ao modo de permissões padrão na Amazon QLDB

Use esta seção para começar a usar o modo de permissões padrão na AmazonQLDB. Esta seção fornece uma tabela de referência para ajudá-lo a escrever uma política baseada em identidade no AWS Identity and Access Management (IAM) para ações partiQL e recursos de tabela em. QLDB Também inclui um step-by-step tutorial para criar políticas de permissões e instruções para encontrar uma tabela ARN e criar tags de tabela emQLDB. IAM

Tópicos

- O modo de permissões STANDARD
- Referência de permissões PartiQL

- Encontrar um ID de tabela e ARN
- Tabelas de marcação
- Tutorial de início rápido: Criação de políticas de permissões

O modo de permissões **STANDARD**

QLDBagora suporta um modo de `STANDARD` permissões para recursos contábeis. O modo de permissões que permite o controle do acesso com granularidade mais fina para ledgers, tabelas e comandos PartiQL. Por padrão, esse modo nega todas as solicitações do usuário para executar comandos do PartiQL em qualquer tabela nesse ledger.

> **Note**
>
> Anteriormente, o único modo de permissões disponível para um ledger era `ALLOW_ALL`. O `ALLOW_ALL` modo permite controle de acesso com granularidade API de nível para livros contábeis e continua sendo suportado, mas não é recomendado, para livros contábeis. QLDB Esse modo permite que os usuários com `SendCommand` API permissão executem todos os comandos do PartiQL em qualquer tabela no livro contábil especificada pela política de permissões (portanto, “permitir todos” os comandos do PartiQL).
> Você pode alterar o modo de permissões dos ledgers existentes de `ALLOW_ALL` para `STANDARD`. Para ter mais informações, consulte Migrar para o modo de permissões padrão.

Para permitir comandos no modo padrão, você deve criar uma política de permissões IAM para recursos de tabela específicos e ações do PartiQL. Isso é um acréscimo à `SendCommand` API permissão para o livro contábil. Para facilitar as políticas nesse modo, QLDB introduziu um conjunto de IAM ações para comandos PartiQL e Amazon Resource Names (ARNs) para QLDB tabelas. Para obter mais informações sobre o modelo QLDB de objeto de dados, consulteConceitos e terminologia fundamentais na Amazon QLDB.

Referência de permissões PartiQL

A tabela a seguir lista cada comando do QLDB PartiQL, as IAM ações correspondentes para as quais você deve conceder permissões para executar o comando e as AWS recursos para os quais você pode conceder as permissões. Você especifica as ações no campo `Action` da política e o valor do recurso no campo `Resource` da política.

> **Important**
>
> - IAMas políticas que concedem permissões a esses comandos do PartiQL só se aplicam ao seu livro contábil se o modo de `STANDARD` permissões estiver atribuído ao livro contábil. Essas políticas não são aplicáveis aos ledgers no modo de permissões `ALLOW_ALL`.
>
> Para saber como especificar o modo de permissões ao criar ou atualizar um ledger, consulte Operações básicas para livros contábeis da Amazon QLDB ou Etapa 1: criar um novo ledger em Conceitos básicos do console.
>
> - Para executar qualquer comando do PartiQL em um livro contábil, você também deve conceder permissão para a `SendCommand` API ação do recurso contábil. Isso é um acréscimo às ações do PartiQL e aos recursos da tabela que estão listados na tabela a seguir. Para obter mais informações, consulte Executar transações de dados.

Comandos do Amazon QLDB PartiQL e permissões necessárias

Comando	Permissões necessárias (IAMações)	Recursos	Ações dependentes
CREATE TABLE	`qldb:PartiQLCreateTable`	`arn:aws:qldb: region:account-id :ledger/ledger-name /table/*`	`qldb:TagResource` (para marcar na criação)
DROP TABLE	`qldb:PartiQLDropTable`	`arn:aws:qldb: region:account-id :ledger/ledger-name /table/table-id`	

Comando	Permissões necessárias (IAMações)	Recursos	Ações dependentes
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 (para documentos inteiros)	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*	
UPDATE FROM(INSERT,REMOV, ouSET)	qldb:PartiQLUpdate	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect

Comando	Permissões necessárias (IAMações)	Recursos	Ações dependentes
REDACT_REVISION (procedimento armazenado)	qldb:PartiQLRedact	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SELECTFROMnome_tabela	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SELECTFROMesquema_de_informações.tabelas_de_usuários	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables	
SELECTFROMhistórico (nome_tabela)	qldb:PartiQLHistoryFunction	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Para obter exemplos de documentos de IAM política que concedem permissões para esses comandos do PartiQL, acesse Tutorial de início rápido: Criação de políticas de permissões ou consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Encontrar um ID de tabela e ARN

Você pode encontrar um ID de tabela usando o AWS Management Console ou consultando a tabela information_schema.user_tables. Para visualizar os detalhes da tabela no console ou consultar essa tabela do catálogo do sistema, você deve ter permissão `SELECT` no recurso do catálogo do sistema. Por exemplo, para encontrar o ID da tabela `Vehicle`, você pode executar a seguinte instrução.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

Esta consulta devolve resultados em um formato semelhante ao seguinte exemplo:

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

Para conceder permissões para executar instruções partiQL em uma tabela, você especifica um recurso de tabela no formato a seguirARN.

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

Veja a seguir um exemplo de tabela ARN para ID de tabela`Au1EiThbt8s0z9wM26REZN`.

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

Usar o console

Você também pode usar o QLDB console para encontrar uma mesaARN.

Para encontrar o ARN de uma mesa (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.

2. No painel de navegação, escolha Ledgers.
3. Na lista de livros contábeis, escolha o nome do livro cuja tabela ARN você deseja encontrar.
4. Na página de detalhes do livro contábil, na guia Tabelas, localize o nome da tabela ARN que você deseja encontrar. Para copiar oARN, escolha o ícone de cópia (⧉) ao lado dele.

Tabelas de marcação

Você já pode marcar os recursos da tabela. Para gerenciar tags para tabelas existentes, use o AWS Management Console ou as API operações `TagResourceUntagResource`, `ListTagsForResource` e. Para obter mais informações, consulte Marcação de recursos da Amazon QLDB.

> **Note**
>
> Os recursos de tabela não herdam as tags de seu recurso de ledger raiz. Atualmente, a marcação de tabelas na criação é suportada apenas para ledgers no modo de permissões `STANDARD`.

Você também pode definir tags de tabela ao criar a tabela usando o QLDB console ou especificando-as em uma instrução `CREATE TABLE` partiQL. O exemplo a seguir cria uma tabela chamada `Vehicle` com a tag `environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Marcar tabelas na criação requer acesso às ações `qldb:PartiQLCreateTable` e `qldb:TagResource`.

Ao marcar os recursos no momento da criação, você elimina a necessidade de executar scripts personalizados de marcação após a criação do recurso. Depois que uma tabela é marcada, você pode controlar o acesso à tabela com base nessas tags. Por exemplo, você pode conceder acesso total somente a tabelas que tenham uma tag específica. Para obter um exemplo JSON de política, consulteAcesso total a todas as ações com base nas tags da tabela.

Usar o console

Você também pode usar o QLDB console para definir as tags da tabela enquanto cria a tabela.

Para marcar uma tabela na criação (console)

1. Faça login no AWS Management Console e abra o QLDB console da Amazon em https://console.aws.amazon.com/qldb.
2. No painel de navegação, escolha Ledgers.
3. Na lista de Ledgers, escolha o nome do livro no qual você deseja criar a tabela.
4. Na página de detalhes do ledger, na guia Tabelas, escolha Criar tabela.
5. Na página Criar tabela, faça o seguinte:
   - Nome da tabela: escolha um nome da tabela.
   - Tags: adicione metadados à tabela anexando tags como pares chave-valor. Você pode adicionar tags à sua tabela para ajudar a organizá-las e identificá-las.

     Escolha Adicionar tag e, em seguida, insira os pares de valores-chave, conforme apropriado.
6. Quando estiver de acordo com as configurações, escolha Create table (Criar tabela).

Tutorial de início rápido: Criação de políticas de permissões

Este tutorial orienta você pelas etapas para criar políticas de permissões em IAM um QLDB livro contábil da Amazon no modo de `STANDARD` permissões. Em seguida, você pode atribuir as permissões aos seus usuários, grupos ou perfis.

Para obter mais exemplos de documentos de IAM política que concedem permissões aos comandos e recursos de tabela do PartiQL, consulte. Exemplos de políticas baseadas em identidade para a Amazon QLDB

Tópicos

- Pré-requisitos
- Criar uma política somente de leitura
- Criar uma política de acesso total
- Criação de uma política somente para leitura para uma tabela específica
- Atribuindo permissões

Pré-requisitos

Antes de iniciar, certifique-se de fazer o seguinte:

1. Siga o AWS instruções de configuração emAcessando a Amazon QLDB, se você ainda não tiver feito isso. Essas etapas incluem a inscrição no AWS e criando um usuário administrativo.
2. Crie um novo ledger e escolha o modo de `STANDARD` permissões para o ledger. Para saber como, consulte Etapa 1: criar um novo ledger em Conceitos básicos do console ou Operações básicas para livros contábeis da Amazon QLDB.

Criar uma política somente de leitura

Para usar o editor JSON de políticas para criar uma política somente para leitura para todas as tabelas em um livro contábil no modo de permissões padrão, faça o seguinte:

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
2. Na coluna de navegação à esquerda, selecione Políticas.

   Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.
3. Na parte superior da página, escolha Criar política.
4. Escolha a JSONguia.
5. Copie e cole o seguinte documento JSON de política. Este exemplo de política concede acesso somente para leitura a todas as tabelas em um ledger.

   Para usar essa política, substitua *us-east-1*, *123456789012* e *myExampleLedger* no exemplo com suas próprias informações.

```
{
  "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. Escolha Revisar política.

> **Note**
>
> Você pode alternar entre o editor visual e JSONas guias a qualquer momento. No entanto, se você fizer alterações ou escolher Revisar política na guia Editor visual, IAM poderá reestruturar sua política para otimizá-la para o editor visual. Para obter mais informações, consulte Reestruturação de políticas no Guia do IAM usuário.

7. Na página Review policy (Revisar política), insira um Name (Nome) e uma Description (Descrição) opcional para a política que você está criando. Revise o Resumo da política para ver as permissões que são concedidas pela política. Em seguida, escolha Criar política para salvar seu trabalho.

Criar uma política de acesso total

Para criar uma política de acesso total para todas as tabelas em um QLDB livro contábil no modo de permissões padrão, faça o seguinte:

- Repita as etapas anteriores usando o seguinte documento de política. Este exemplo de política concede acesso a todos os comandos do PartiQL para todas as tabelas em um ledger, usando curingas (*) para cobrir todas as ações do PartiQL e todos os recursos em um ledger.

> **Warning**
>
> Este é um exemplo do uso de um caractere curinga (*) para permitir todas as ações do PartiQL, incluindo operações administrativas e de leitura/gravação em todas as tabelas em um livro contábil. QLDB Em vez disso, é uma prática recomendada especificar explicitamente cada ação a ser concedida, e apenas o que esse usuário, função ou grupo precisa.

Para usar essa política, substitua *us-east-1*, *123456789012* e *myExampleLedger* no exemplo com suas próprias informações.

```
{
  "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"
      ]
    }
  ]
}
```

Criação de uma política somente para leitura para uma tabela específica

Para criar uma política de acesso somente para leitura para uma tabela específica em um QLDB livro contábil no modo de permissões padrão, faça o seguinte:

1. Encontre o ARN para a tabela usando o AWS Management Console ou consultando a tabela `information_schema.user_tables` do catálogo do sistema. Para obter instruções, consulte Encontrar um ID de tabela e ARN.
2. Use a tabela ARN para criar uma política que permita acesso somente para leitura à tabela. Para fazer isso, repita as etapas anteriores usando o documento de política a seguir.

   Esse exemplo de política concede acesso somente para leitura à tabela especificada. Nesse exemplo, o ID da tabela é `Au1EiThbt8s0z9wM26REZN`. Para usar essa política, substitua *`us-east-1`*, *`123456789012`*, *`myExampleLedger`* e *`Au1EiThbt8s0z9wM26REZN`* no exemplo com suas próprias informações.

```
{
  "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"
      ]
    }
  ]
}
```

Atribuindo permissões

Depois de criar uma política de QLDB permissões, você atribui as permissões da seguinte forma.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criar um conjunto de permissões no AWS IAM Identity Center Guia do usuário.
- Usuários gerenciados IAM por meio de um provedor de identidade:

  Crie um perfil para a federação de identidades. Siga as instruções em Criação de uma função para um provedor de identidade terceirizado (federação) no Guia IAM do usuário.
- IAMusuários:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de uma função para um IAM usuário no Guia IAM do usuário.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adicionar permissões a um usuário (console) no Guia do IAM usuário.

Exemplos de políticas baseadas em identidade para a Amazon QLDB

Por padrão, usuários e funções não têm permissão para criar ou modificar QLDB recursos. Eles também não podem realizar tarefas usando o AWS Management Console, AWS Command Line Interface (AWS CLI), ou AWS API. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

Para saber como criar uma política IAM baseada em identidade usando esses exemplos de documentos de JSON política, consulte Criação de IAM políticas no Guia do IAMusuário.

Para obter detalhes sobre ações e tipos de recursos definidos porQLDB, incluindo o formato de cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição para a Amazon QLDB na Referência de Autorização de Serviço. ARNs

Sumário

- Melhores práticas de política

- Usando o QLDB console
  - Permissões do histórico de consultas
  - Permissões completas do console sem histórico de consultas
- Permitir que usuários visualizem suas próprias permissões
- Executar transações de dados
  - Permissões padrão para ações do PartiQL e recursos de tabela
    - Permitir acesso total a todas as ações
    - Acesso total a todas as ações com base nas tags da tabela
    - Acesso de leitura/gravação
    - Acesso somente leitura
    - Acesso somente leitura a uma tabela específica
    - Permitir acesso para criar tabelas
    - Permitir acesso para criar tabelas com base nas tags de solicitação
- Exportar um diário para um bucket do Amazon S3
- Fluxo de um diário para o Kinesis Data Streams
- Atualizando QLDB livros contábeis com base em etiquetas

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir QLDB recursos em sua conta. Essas ações podem incorrer em custos para o seu Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com AWS políticas gerenciadas e migrar para permissões com privilégios mínimos — Para começar a conceder permissões para seus usuários e cargas de trabalho, use o AWS políticas gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis em seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões

definindo AWS políticas gerenciadas pelo cliente que são específicas para seus casos de uso. Para ter mais informações, consulte AWS políticas gerenciadas ou AWS políticas gerenciadas para funções de trabalho no Guia IAM do usuário.

- Aplique permissões com privilégios mínimos — Ao definir permissões com IAM políticas, conceda somente as permissões necessárias para realizar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre IAM como usar para aplicar permissões, consulte Políticas e permissões IAM no Guia IAM do usuário.
- Use condições nas IAM políticas para restringir ainda mais o acesso — Você pode adicionar uma condição às suas políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usandoSSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de um determinado AWS service (Serviço da AWS), por exemplo, AWS CloudFormation. Para obter mais informações, consulte elementos IAM JSON da política: Condição no Guia IAM do usuário.
- Use o IAM Access Analyzer para validar suas IAM políticas e garantir permissões seguras e funcionais — o IAM Access Analyzer valida políticas novas e existentes para que as políticas sigam a linguagem da IAM política (JSON) e as melhores práticas. IAM IAMO Access Analyzer fornece mais de 100 verificações de políticas e recomendações práticas para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação da política do IAM Access Analyzer no Guia do IAM Usuário.
- Exigir autenticação multifatorial (MFA) — Se você tiver um cenário que exija IAM usuários ou um usuário root em seu Conta da AWS, ative MFA para obter segurança adicional. Para exigir MFA quando API as operações são chamadas, adicione MFA condições às suas políticas. Para obter mais informações, consulte Configurando o API acesso MFA protegido no Guia do IAMusuário.

Para obter mais informações sobre as melhores práticas emIAM, consulte as melhores práticas de segurança IAM no Guia IAM do usuário.

Usando o QLDB console

Para acessar o QLDB console da Amazon, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os QLDB recursos em seu Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas exigidas, o console não funcionará conforme planejado para entidades (usuários ou funções) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para o AWS CLI ou o AWS API. Em vez disso, permita o acesso somente às ações que correspondam à API operação que eles estão tentando realizar.

Para garantir que os usuários e as funções tenham acesso total ao QLDB console e a todos os seus recursos, anexe o seguinte AWS política gerenciada para as entidades. Para obter mais informaçõesAWS políticas gerenciadas para a Amazon QLDB, consulte Adicionar permissões a um usuário no Guia do IAM usuário.

```
AmazonQLDBConsoleFullAccess
```

Permissões do histórico de consultas

Além das QLDB permissões, alguns recursos do console exigem permissões para o Database Query Metadata Service (prefixo do serviço:`dbqms`). Este é um serviço somente interno que gerencia suas consultas recentes e salvas no console, editor de consultas para e outros QLDB Serviços da AWS. Para obter uma lista completa de DBQMS API ações, consulte Database Query Metadata Service na Referência de Autorização de Serviço.

Para permitir permissões do histórico de consultas, você pode usar o AWS política gerenciada mazonQLDBConsole FullAccess A. Essa política usa um caractere curinga (`dbqms:*`) para permitir todas as DBQMS ações para todos os recursos.

Ou você pode criar uma IAM política personalizada e incluir as seguintes DBQMS ações. O editor de consultas partiQL no QLDB console exige permissões para usar essas ações nos recursos do histórico de consultas.

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

Permissões completas do console sem histórico de consultas

Para permitir acesso total ao QLDB console sem nenhuma permissão de histórico de consultas, você pode criar uma IAM política personalizada que exclua todas as DBQMS ações. Por exemplo,

o documento de política a seguir permite as mesmas permissões concedidas pelo AWS política gerenciada A mazonQLDBConsole FullAccess, exceto ações que começam com o prefixo `dbqms` do serviço.

```
{
  "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"
        }
      }
    }
  ]
}
```

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permita IAM aos usuários visualizar as políticas embutidas e gerenciadas que estão anexadas à identidade do usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando o AWS CLI ou 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": "*"
        }
    ]
}
```

Executar transações de dados

Para interagir com os dados QLDB transacionais API (QLDBSessão) executando instruções partiQL em um livro contábil, você deve conceder permissão para a ação. `SendCommand` API O JSON documento a seguir é um exemplo de uma política que concede permissão somente para a `SendCommand` API ação no livro contábil`myExampleLedger`.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`* e *`myExampleLedger`* no exemplo com suas próprias informações.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
```

```
}
```

Se `myExampleLedger` usar o modo de `ALLOW_ALL` permissões, essa política concederá permissões para executar todos os comandos do PartiQL em qualquer tabela no ledger.

Você também pode usar um AWS política gerenciada para conceder acesso total a todos os QLDB recursos. Para obter mais informações, consulte AWS políticas gerenciadas para a Amazon QLDB.

Permissões padrão para ações do PartiQL e recursos de tabela

Para livros contábeis no modo de `STANDARD` permissões, você pode consultar os seguintes documentos de IAM política como exemplos de concessão das permissões partiQL apropriadas. Para obter uma lista das permissões necessárias para cada comando do PartiQL, consulte as Referência de permissões PartiQL.

Tópicos

- Permitir acesso total a todas as ações
- Acesso total a todas as ações com base nas tags da tabela
- Acesso de leitura/gravação
- Acesso somente leitura
- Acesso somente leitura a uma tabela específica
- Permitir acesso para criar tabelas
- Permitir acesso para criar tabelas com base nas tags de solicitação

Permitir acesso total a todas as ações

O documento JSON de política a seguir concede acesso total para usar todos os comandos do PartiQL em todas as tabelas do. `myExampleLedger` Essa política produz o mesmo efeito do uso do modo de `ALLOW_ALL` permissões para o ledger.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`* e *`myExampleLedger`* no exemplo com suas próprias informações.

```
{
    "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"
            ]
        }
    ]
}
```

Acesso total a todas as ações com base nas tags da tabela

O documento JSON de política a seguir usa uma condição baseada nas tags de recursos da tabela para conceder acesso total ao uso de todos os comandos do PartiQL em todas as tabelas do. `myExampleLedger` As permissões são concedidas somente se a tag da tabela `environment` tiver o valor `development`.

> **Warning**
>
> Este é um exemplo do uso de um caractere curinga (*) para permitir todas as ações do PartiQL, incluindo operações administrativas e de leitura/gravação em todas as tabelas em um livro contábil. QLDB Em vez disso, é uma prática recomendada especificar explicitamente cada ação a ser concedida, e apenas o que esse usuário, função ou grupo precisa.

Para usar essa política, substitua *us-east-1*, *123456789012* e *myExampleLedger* no exemplo com suas próprias informações.

```
{
    "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" }
            }
        }
    ]
}
```

Acesso de leitura/gravação

O documento JSON de política a seguir concede permissões para selecionar, inserir, atualizar e excluir dados em todas as tabelas domy`ExampleLedger`. Essa política não concede permissões para redigir dados ou alterar o esquema, por exemplo, criar e eliminar tabelas e índices.

> **Note**
>
> Uma `UPDATE` declaração exige permissões para as ações `qldb:PartiQLUpdate` e `qldb:PartiQLSelect` da tabela que está sendo modificada. Quando você executa uma instrução `UPDATE`, ela executa uma operação de leitura além da operação de atualização. A

exigência de ambas as ações garante que somente os usuários que têm permissão para ler o conteúdo de uma tabela recebam permissões `UPDATE`.
Da mesma forma, uma declaração `DELETE` exige permissões para as ações `qldb:PartiQLDelete` e `qldb:PartiQLSelect`.

Para usar essa política, substitua *us-east-1*, *123456789012* e *myExampleLedger* no exemplo com suas próprias informações.

```
{
    "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"
            ]
        }
    ]
}
```

Acesso somente leitura

O documento JSON de política a seguir concede permissões somente para leitura em todas as tabelas do. `myExampleLedger` Para usar essa política, substitua *us-east-1*, *123456789012* e *myExampleLedger* no exemplo com suas próprias informações.

```
{
    "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"
            ]
        }
    ]
}
```

Acesso somente leitura a uma tabela específica

O documento JSON de política a seguir concede permissões somente para leitura em uma tabela específica em. `myExampleLedger` Nesse exemplo, o ID da tabela é `Au1EiThbt8s0z9wM26REZN`.

Para usar essa política, substitua *us-east-1*, *123456789012*, *myExampleLedger* e *Au1EiThbt8s0z9wM26REZN* no exemplo com suas próprias informações.

```
{
    "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"
            ]
        }
    ]
}
```

Permitir acesso para criar tabelas

O documento JSON de política a seguir concede permissão para criar tabelas emmy`ExampleLedger`. A ação `qldb:PartiQLCreateTable` requer permissões para o tipo de recurso da tabela. No entanto, o ID da tabela nova não é conhecido no momento em que você executa uma instrução `CREATE TABLE`. Portanto, uma política que concede a `qldb:PartiQLCreateTable` permissão deve usar um caractere curinga (*) na tabela ARN para especificar o recurso.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`* e *`myExampleLedger`* no exemplo com suas próprias informações.

```
{
    "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/*"
            ]
        }
    ]
}
```

Permitir acesso para criar tabelas com base nas tags de solicitação

O documento JSON de política a seguir usa uma condição baseada na chave de `aws:RequestTag` contexto para conceder permissão para criar tabelas em`myExampleLedger`. As permissões são concedidas somente se a tag de solicitação `environment` tiver o valor `development`. Marcar tabelas na criação requer acesso às ações `qldb:PartiQLCreateTable` e `qldb:TagResource`. Para saber como marcar tabelas na criação de tags, consulte Tabelas de marcação.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`* e *`myExampleLedger`* no exemplo com suas próprias informações.

```
{
    "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" }
            }
        }
    ]
}
```

Exportar um diário para um bucket do Amazon S3

Etapa 1: permissões de exportação do QLDB diário

No exemplo a seguir, você concede a um usuário em seu Conta da AWS permissões para realizar a `qldb:ExportJournalToS3` ação em um recurso QLDB contábil. Você também concede permissões para realizar a `iam:PassRole` ação no recurso de IAM função que deseja passar para o QLDB serviço. Isso é necessário para todas as solicitações de exportação de diário.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`*, *`myExampleLedger`* e *`qldb-s3-export`* no exemplo com suas próprias informações.

```
{
    "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"
               }
            }
        }
    ]
}
```

Etapa 2: Permissões do Amazon S3 bucket

No exemplo a seguir, você usa uma IAM função para conceder QLDB acesso à gravação em um dos seus buckets do Amazon S3,. `amzn-s3-demo-bucket` Isso também é necessário para todas as exportações de QLDB periódicos.

Além de conceder a `s3:PutObject` permissão, a política também concede a `s3:PutObjectAcl` permissão para definir as permissões da lista de controle de acesso (ACL) para um objeto.

Para usar essa política, substitua amzn-s3-demo-bucket no exemplo pelo nome do seu bucket Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Em seguida, você anexa essa política de permissões a uma IAM função que QLDB pode assumir para acessar seu bucket do Amazon S3. O JSON documento a seguir é um exemplo de uma política de confiança que QLDB permite assumir a IAM função de qualquer QLDB recurso `123456789012` somente na conta.

Para usar essa política, substitua *`us-east-1`* e *`123456789012`* no exemplo com suas próprias informações.

```
{
    "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"
                }
            }
        }
    ]
}
```

Fluxo de um diário para o Kinesis Data Streams

Etapa 1: permissões do stream do QLDB diário

No exemplo a seguir, você concede a um usuário em seu Conta da AWS permissões para realizar a `qldb:StreamJournalToKinesis` ação em todos os sub-recursos do QLDB stream em um livro contábil. Você também concede permissões para realizar a `iam:PassRole` ação no recurso de IAM função que deseja passar para o QLDB serviço. Isso é necessário para todas as solicitações de fluxos de diário.

Para usar essa política, substitua *us-east-1*, *123456789012*, *myExampleLedger* e *qldb-kinesis-stream* no exemplo com suas próprias informações.

```
{
    "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"
               }
            }
        }
    ]
```

```
}
```

Etapa 2: permissões do Kinesis Data Streams

No exemplo a seguir, você usa uma IAM função para conceder QLDB acesso para gravar registros de dados em seu stream de dados do Amazon Kinesis, *stream-for-qldb*. Isso também é necessário para todos os fluxos de QLDB diários.

Para usar essa política, substitua *us-east-1*, *123456789012* e *stream-for-qldb* no exemplo com suas próprias informações.

```
{
    "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"
        }
    ]
}
```

Em seguida, você anexa essa política de permissões a uma IAM função que QLDB pode assumir para acessar seu stream de dados do Kinesis. O JSON documento a seguir é um exemplo de uma política de confiança que QLDB permite assumir uma IAM função em qualquer QLDB fluxo na conta `myExampleLedger` somente `123456789012` para o livro contábil.

Para usar essa política, substitua *us-east-1*, *123456789012* e *myExampleLedger* no exemplo com suas próprias informações.

```
{
    "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"
                }
            }
        }
    ]
}
```

Atualizando QLDB livros contábeis com base em etiquetas

Você pode usar condições em sua política baseada em identidade para controlar o acesso aos QLDB recursos com base em tags. Este exemplo mostra como é possível criar uma política que permite atualizar um ledger. No entanto, a permissão é concedida somente se a tag do ledger `Owner` tiver o valor do nome de usuário desse usuário. Essa política também concede as permissões necessárias concluir essa ação no console.

```
{
    "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}"}
            }
        }
    ]
}
```

Você pode anexar essa política aos usuários na sua conta. Se um usuário chamado `richard-roe` tentar atualizar um QLDB livro contábil, o registro deverá estar marcado com `Owner=richard-roe` ou. `owner=richard-roe` Caso contrário, ele terá o acesso negado. A chave da tag de condição `Owner` corresponde a `Owner` e a `owner` porque os nomes de chaves de condição não diferenciam letras maiúsculas de minúsculas. Para obter mais informações, consulte elementos IAM JSON da política: Condição no Guia IAM do usuário.

Prevenção do problema do substituto confuso entre serviços

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O problema de "confused deputy" é uma questão de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Em AWS, a falsificação de identidade entre serviços pode resultar no problema confuso do deputado.

A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar o confuso problema do deputado, AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com diretores de serviços que receberam acesso aos recursos em sua conta.

Recomendamos usar as chaves `aws:SourceArn`de contexto de condição `aws:SourceAccount`global nas políticas de recursos para limitar as permissões que a Amazon QLDB concede a outro serviço ao recurso. Se você utilizar ambas as chaves de contexto de condição global, o valor `aws:SourceAccount` e a conta no valor `aws:SourceArn` deverão utilizar o mesmo ID de conta quando utilizados na mesma instrução de política.

A tabela a seguir lista os valores possíveis `StreamsJournalToKinesis`QLDBAPIdas operações `ExportJournalToS3`e. `aws:SourceArn` Essas operações estão no escopo desse problema de segurança porque elas chamam AWS Security Token Service (AWS STS) para assumir uma IAM função que você especificar.

APIoperação	Serviço chamado	leis: SourceArn
`ExportJournalToS3`	AWS STS (AssumeRole)	Permite QLDB assumir a função de qualquer QLDB recurso na conta: `arn:aws:qldb: us-east-1 :123456789012 :*` Atualmente, QLDB só suporta esse caractere curinga ARN para exportações de periódicos.

`StreamsJournalToKinesis`	AWS STS (AssumeRole)	Permite QLDB assumir a função de um QLDB fluxo específico:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		Observação: você só pode especificar um ID de stream ARN após a criação do recurso de stream. Usando issoARN, você pode permitir que a função seja usada somente para um único QLDB stream.
		Permite assumir QLDB a função de qualquer QLDB fluxo de um livro contábil:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`
		Permite QLDB assumir a função de qualquer QLDB fluxo na conta:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		Permite QLDB assumir a função de qualquer QLDB recurso na conta:
		`arn:aws:qldb: us-east-1 :123456789012 :*`

A maneira mais eficaz de se proteger contra o confuso problema do deputado é usar a chave de contexto ARN de condição `aws:SourceArn` global com todo o recurso. Se você não souber a totalidade ARN do recurso ou se estiver especificando vários recursos, use a chave de condição de contexto `aws:SourceArn` global com caracteres curinga (`*`) para as partes desconhecidas do ARN —por exemplo,. `arn:aws:qldb:us-east-1:`*`123456789012`*`:*`

O exemplo de política de confiança a seguir para uma IAM função mostra como você pode usar as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto para evitar o confuso problema adjunto. Com essa política de confiança, QLDB pode assumir a função de qualquer QLDB fluxo na conta `myExampleLedger` somente `123456789012` para o livro contábil.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`* e *`myExampleLedger`* no exemplo com suas próprias informações.

```
{
  "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 políticas gerenciadas para a Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Uma AWS a política gerenciada é uma política autônoma criada e administrada pelo AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Tenha em mente que AWS políticas gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para todos AWS clientes para usar. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da específicas para seus casos de uso.

Você não pode alterar as permissões definidas em AWS políticas gerenciadas. If (Se) AWS atualiza as permissões definidas em um AWS política gerenciada, a atualização afeta todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize um AWS política gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas API operações são disponibilizadas para os serviços existentes.

Para ter mais informações, consulte AWS políticas gerenciadas no Guia IAM do usuário.

Para obter mais informações sobre as QLDB API operações nesses AWS políticas gerenciadas, consulte QLDBAPIReferência da Amazon o.

Tópicos

- AWS política gerenciada: mazonQLDBRead Somente A
- AWS política gerenciada: A mazonQLDBFull Access
- AWS política gerenciada: A mazonQLDBConsole FullAccess
- QLDBatualizações para AWS políticas gerenciadas

AWS política gerenciada: mazonQLDBRead Somente A

Use a política A mazonQLDBRead Only para conceder permissões de somente leitura a todos os QLDB recursos. Você pode anexar essa política às suas IAM identidades.

Detalhes da permissão

Esta política inclui as seguintes permissões para o serviço `qldb`.

- Permite que os diretores descrevam e listem todos os QLDB recursos e suas tags. Esses recursos incluem livros contábeis, trabalhos de exportação do Amazon S3 e streams para o Kinesis Data Streams.
- Permite que as entidades principais obtenham um bloco, resumo ou revisão do diário em qualquer livro contábil para verificar os dados criptograficamente.
- Não permite que as entidades principais executem comandos do PartiQL em nenhuma tabela em nenhum ledger.

AWS política gerenciada: A mazonQLDBFull Access

Use a política A mazonQLDBFull Access para conceder permissões administrativas completas a todos os QLDB recursos por meio do QLDB API ou do AWS CLI. Você pode anexar essa política às suas IAM identidades.

Detalhes das permissões

Esta política inclui as seguintes permissões:

- `qldb`
  - Permite que os diretores criem, descrevam, listem e gerenciem todos os QLDB recursos e suas tags. Esses recursos incluem livros contábeis, trabalhos de exportação do Amazon S3 e streams para o Kinesis Data Streams.
  - Permite que os diretores executem todos os comandos do PartiQL em todas as tabelas em qualquer livro contábil usando QLDBo driver ou o shell. QLDB
  - Permite que as entidades principais obtenham um bloco, resumo ou revisão do diário em qualquer livro contábil para verificar os dados criptograficamente.

- `iam`— Permite que os diretores passem qualquer recurso de IAM função em sua conta para o QLDB serviço. Isso é necessário para todas as solicitações de streams de diário.

AWS política gerenciada: A mazonQLDBConsole FullAccess

Use a mazonQLDBConsole FullAccess política A para conceder permissões administrativas completas a todos os QLDB recursos por meio do AWS Management Console QLDBAPI, o ou o AWS CLI. Você pode anexar essa política às suas IAM identidades.

Detalhes das permissões

Esta política inclui as seguintes permissões:

- `qldb`
  - Permite que os diretores criem, descrevam, listem e gerenciem todos os QLDB recursos e suas tags. Esses recursos incluem livros contábeis, trabalhos de exportação do Amazon S3 e streams para o Kinesis Data Streams.
  - Permite que os diretores executem todos os comandos do PartiQL em todas as tabelas em qualquer livro contábil usando QLDB o console, o driver ou QLDBo shell. QLDB
  - Permite que os diretores insiram dados de exemplo do aplicativo em qualquer livro contábil usando o QLDB console.
  - Permite que as entidades principais obtenham um bloco, resumo ou revisão do diário em qualquer livro contábil para verificar os dados criptograficamente.
- `dbqms`: permite que as entidades principais usem todas as ações no Database Query Metadata Service. Esse é um serviço somente interno que o QLDB console exige para criar, descrever e gerenciar consultas recentes e salvas para o editor de consultas partiQL.
- `kinesis`: permite que as entidades principais descrevam e listem recursos do Amazon Kinesis Data Streams. Esses recursos são os destinos de destino nos quais os recursos de QLDB streaming podem gravar dados.
- `iam`— Permite que os diretores passem qualquer recurso de IAM função em sua conta para o QLDB serviço. Isso é necessário para todas as solicitações de streams de diário.

QLDBatualizações para AWS políticas gerenciadas

Exibir detalhes sobre as atualizações do AWS políticas gerenciadas QLDB desde que esse serviço começou a rastrear essas mudanças. Para receber alertas automáticos sobre alterações nessa página, assine o RSS feed na página Histórico de QLDB lançamentos.

Alteração	Descrição	Data
A mazonQLDBFull Access, A mazonQLDBConsole FullAccess — Atualização das políticas existentes	QLDBadicionou uma nova permissão para permitir que os diretores editem revisões de documentos em todos os livros contábeis no modo de permissões. `STANDARD`	4 de novembro de 2022
A mazonQLDBFull Access, A mazonQLDBConsole FullAccess — Atualização das políticas existentes	QLDBadicionou novas permissões para permitir que os diretores passem qualquer recurso de IAM função em sua conta para o QLDB serviço. Isso é necessário para todas as solicitações de streams de diário.	2 de setembro de 2021
mazonQLDBReadSomente A — Atualização de uma política existente	QLDBremoveu uma `qldb:GetBlock` ação duplicada que foi listada anteriormente duas vezes e reordenou o `"Effect"` campo para que ele apareça antes do `"Action"` campo.	1.º de julho de 2021
A mazonQLDBFull Access, A mazonQLDBConsole FullAccess — Atualização das políticas existentes	QLDBadicionou novas permissões para permitir que os diretores atualizem o modo de permissões em todos os livros contábeis e executem todos os comandos do PartiQL em todos os livros	27 de maio de 2021

Alteração	Descrição	Data
	contábeis no novo modo de permissões. `STANDARD` O modo de permissões `STANDARD` suporta controle de acesso em nível de tabela e granularidade para comandos partiQL. Para facilitar o novo modo de permissões, QLDB introduziu um conjunto de IAM ações para tipos de comando PartiQL e Amazon Resource Names (ARNs) para recursos de QLDB tabela. Essas duas políticas foram atualizadas para incluir as novas ações do PartiQL para conceder acesso total aos ledgers `STANDARD`.	
O QLDB iniciou o rastreamento das alterações	QLDBcomeçou a rastrear as mudanças em seu AWS políticas gerenciadas.	1.º de março de 2021

Solução de problemas de QLDB identidade e acesso da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com QLDB e. IAM

Tópicos

- Não estou autorizado a realizar uma ação em QLDB
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha Conta da AWS para acessar meus QLDB recursos

Não estou autorizado a realizar uma ação em QLDB

Se o AWS Management Console informa que você não está autorizado a realizar uma ação e, em seguida, entre em contato com o administrador para obter ajuda. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`myExampleLedger`* fictício, mas não tem as permissões fictícias do `qldb:`*`DescribeLedger`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas e permitir o acesso ao recurso *`myExampleLedger`* usando a ação `qldb:`*`DescribeLedger`*.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a `iam:PassRole` ação, suas políticas devem ser atualizadas para permitir que você transfira uma função para QLDB o.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um IAM usuário chamado `marymajor` tenta usar o console para realizar uma ação noQLDB. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Para obter orientação sobre solução de problemas sobre esse erro específico das operações de exportação ou transmissão do diário, consulte Solução de problemas da Amazon QLDB.

Quero permitir que pessoas fora da minha Conta da AWS para acessar meus QLDB recursos

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é QLDB compatível com esses recursos, consulteComo a Amazon QLDB trabalha com IAM.
- Para saber como fornecer acesso aos seus recursos em Contas da AWS que você possui, consulte Fornecendo acesso a um IAM usuário em outro Conta da AWS que você possui no Guia do IAM Usuário.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Fornecendo acesso a Contas da AWS propriedade de terceiros no Guia do IAM Usuário.
- Para saber como fornecer acesso por meio da federação de identidades, consulte Fornecendo acesso a usuários autenticados externamente (federação de identidades) no Guia do IAMusuário.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Registro e monitoramento na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho da Amazon QLDB e de seus AWS soluções. Você deve coletar dados de monitoramento de todas as partes do seu AWS solução para que você possa depurar mais facilmente uma falha de vários pontos, caso ocorra. No entanto, antes de iniciar o monitoramentoQLDB, você deve criar um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer uma linha de base para o QLDB desempenho normal em seu ambiente, medindo o desempenho em vários momentos e sob diferentes condições de carga. Ao monitorarQLDB, armazene dados históricos de monitoramento para que você possa compará-los com os dados de desempenho atuais, identificar padrões normais de desempenho e anomalias de desempenho e criar métodos para resolver problemas.

Para estabelecer uma linha de base, é preciso, no mínimo, monitorar os seguintes itens:

- Leia e grave I/Os e armazenamento, para que você possa rastrear os padrões de consumo do seu ledger para fins de cobrança.
- Latência de comando, para que você possa acompanhar o desempenho do seu ledger ao executar operações de dados.
- Exceções, para que você possa determinar se alguma solicitação resultou em erro.

Tópicos

- Ferramentas de monitoramento
- Monitoramento com a Amazon CloudWatch
- Automatizando a Amazon QLDB com eventos CloudWatch
- Registrando QLDB API chamadas da Amazon com AWS CloudTrail

Ferramentas de monitoramento

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

AWS fornece várias ferramentas que você pode usar para monitorar a AmazonQLDB. É possível configurar algumas dessas ferramentas para fazer o monitoramento em seu lugar, e, ao mesmo tempo, algumas das ferramentas exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Tópicos

- Ferramentas de monitoramento automatizadas
- Ferramentas de monitoramento manual

Ferramentas de monitoramento automatizadas

Você pode usar as seguintes ferramentas de monitoramento automatizado para observar QLDB e relatar quando algo está errado:

- Amazon CloudWatch Alarms — Observe uma única métrica em um período especificado por você e execute uma ou mais ações com base no valor da métrica em relação a um determinado limite em vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (AmazonSNS) ou para uma política do Amazon EC2 Auto Scaling. CloudWatch os alarmes não invocam ações simplesmente porque estão em um determinado estado; o estado deve ter sido alterado e mantido por um determinado número de períodos. Para obter mais informações, consulte Monitoramento com a Amazon CloudWatch.
- Amazon CloudWatch Logs — Monitore, armazene e acesse seus arquivos de log em AWS CloudTrail ou outras fontes. Para obter mais informações, consulte Monitoramento de arquivos de log no Guia CloudWatch do usuário da Amazon.
- Amazon CloudWatch Events — Combine eventos e encaminhe-os para uma ou mais funções ou streams de destino para fazer alterações, capturar informações de estado e tomar medidas corretivas. Para obter mais informações, consulte O que é Amazon CloudWatch Events no Guia CloudWatch do usuário da Amazon.

- AWS CloudTrail Monitoramento de registros — compartilhe arquivos de log entre contas, monitore arquivos de CloudTrail log em tempo real enviando-os para o CloudWatch Logs, grave aplicativos de processamento de log em Java e valide se seus arquivos de log não foram alterados após a entrega. CloudTrail Para obter mais informações, consulte Trabalhando com arquivos de CloudTrail log no AWS CloudTrail Guia do usuário.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento QLDB envolve o monitoramento manual dos itens que os CloudWatch alarmes não cobrem. OQLDB, CloudWatch, Trusted Advisor, e outros AWS Management Console os painéis fornecem uma at-a-glance visão do estado do seu AWS meio ambiente. Recomendamos que você também verifique os arquivos de log na AmazonQLDB.

- O painel do QLDB mostra o seguinte:
  - Ler e gravar I/Os
  - Diário e armazenamento indexado
  - Latência de comando
  - Exceções
- A página CloudWatch inicial mostra o seguinte:
  - Alertas e status atual
  - Gráficos de alertas e recursos
  - Estado de integridade do serviço

  Além disso, você pode usar CloudWatch para fazer o seguinte:
  - Crie painéis personalizados para monitorar os serviços com os quais você se preocupa.
  - Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências
  - Pesquise e navegue em todos os seus AWS métricas de recursos
  - Criar e editar alertas para ser notificado sobre problemas

Monitoramento com a Amazon CloudWatch

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode monitorar a Amazon QLDB usando CloudWatch, que coleta e processa dados brutos da Amazon QLDB em métricas legíveis. near-real-time Essas estatísticas são mantidas por duas semanas, de maneira que você possa acessar informações históricas e ter uma perspectiva melhor do desempenho do aplicativo web ou do serviço. Por padrão, os dados QLDB métricos são enviados automaticamente CloudWatch em períodos de 1 ou 15 minutos. Para obter mais informações, consulte O que são Amazon CloudWatch, Amazon CloudWatch Events e Amazon CloudWatch Logs? no Guia do CloudWatch usuário da Amazon.

Tópicos

- Como faço para usar QLDB métricas?
- QLDBMétricas e dimensões da Amazon
- Criação de CloudWatch alarmes para monitorar a Amazon QLDB

Como faço para usar QLDB métricas?

As métricas relatadas por QLDB fornecem informações que você pode analisar de maneiras diferentes. A lista a seguir mostra alguns usos comuns para as métricas. Essas são sugestões para você começar, e não uma lista abrangente.

- Você pode monitorar `JournalStorage` e `IndexedStorage` durante um período de tempo especificado, rastrear quanto espaço em disco seu ledger está consumindo.
- Você pode monitorar `ReadIOs` e `WriteIOs` durante um período de tempo especificado, rastrear quantas solicitações seu ledger está processando.
- Você pode monitorar `CommandLatency` para monitorar o desempenho do seu ledger para operações de dados e analisar os tipos de comandos que resultam em maior latência.

QLDBMétricas e dimensões da Amazon

Quando você interage com a AmazonQLDB, ela envia as seguintes métricas e dimensões para CloudWatch. As métricas de armazenamento são relatadas a cada 15 minutos, e todas as outras métricas são agregadas e relatadas a cada minuto. Você pode usar os procedimentos a seguir para visualizar as métricas deQLDB.

Para visualizar métricas usando o CloudWatch console

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Se necessário, altere a região da . Na barra de navegação, escolha a região onde está AWS os recursos residem. Para obter mais informações, consulte Regiões e endpoints.
3. No painel de navegação, selecione Métricas.
4. Na guia Todas as métricas, escolha QLDB.

Para visualizar métricas usando o AWS CLI

- Em um prompt de comando, use o seguinte comando.

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch exibe as seguintes métricas paraQLDB.

QLDBDimensões e métricas da Amazon

As métricas e dimensões que a Amazon QLDB envia para a Amazon CloudWatch estão listadas aqui.

QLDBmétricas

Métrica	Descrição
`JournalStorage`	A quantidade total de espaço em disco usada pelo diário do ledger, relatada em intervalos de 15 minutos. O diário

Métrica	Descrição
	contém o histórico completo, imutável e verificável de todas as alterações em seus dados. Unidades: `Bytes` Dimensões: `LedgerName`
`IndexedStorage`	A quantidade total de espaço em disco usada pelas tabelas, índices e histórico indexado do ledger, relatada em intervalos de 15 minutos. O armazenamento indexado consiste em dados do ledger otimizados para consultas de alta performance. O máximo é 16 TiB. Unidades: `Bytes` Dimensões: `LedgerName`
`ReadIOs`	O número de solicitações de E/S de leitura, relatadas em intervalos de um minuto. Isso captura todos os tipos de operações de leitura, incluindo transações de dados, solicitações de verificação, exportações de diários e fluxos de diários. Unidades: `Count` Dimensões: `LedgerName`
`WriteIOs`	O número de solicitações de I/O de gravação, relatado em intervalos de um minuto. Unidades: `Count` Dimensões: `LedgerName`

Métrica	Descrição
`CommandLatency`	A quantidade de tempo gasto para operações de dados, relatada em intervalos de um minuto. Unidades: `Milliseconds` Dimensões: `CommandType, LedgerName`
`IsImpaired`	O sinalizador que indica se um fluxo de diário para o Kinesis Data Streams está comprometido, relatado em intervalos de um minuto. Um valor de 1 indica que o fluxo está em estado comprometido e `0` indica o contrário. Unidades: `Boolean` (0 ou 1) Dimensões: `LedgerName, StreamId`
`OccConflictExceptions`	O número de solicitações QLDB que geram um`OccConflictException`. Para obter informaçõ es sobre o controle otimista de simultaneidade (OCC), consulte. Modelo de QLDB concorrência da Amazon Unidades: `Count`
`Session4xxExceptions`	O número de solicitações QLDB que geram um erro HTTP 4xx. Unidades: `Count`
`Session5xxExceptions`	O número de solicitações QLDB que geram um erro HTTP 5xx. Unidades: `Count`
`SessionRateExceeded Exceptions`	O número de solicitações QLDB que geram uma`SessionRateExceededException`. Unidades: `Count`

Dimensões para métricas do QLDB

As métricas de QLDB são qualificadas pelos valores da conta, nome do livro contábil, ID do fluxo ou tipo de comando. Você pode usar o CloudWatch console para recuperar QLDB dados em qualquer uma das dimensões na tabela a seguir.

Dimensão	Descrição
`LedgerName`	Esta dimensão limita os dados a um ledger específico. Esse valor pode ser qualquer nome de livro contábil no atual Região da AWS e o atual Conta da AWS.
`StreamId`	Esta dimensão limita os dados a um rótulo de fluxo específico. Esse valor pode ser qualquer ID de fluxo para um livro contábil no atual Região da AWS e o atual Conta da AWS.
`CommandType`	Essa dimensão limita os dados a um dos seguintes API comandos QLDB de dados: • `AbortTransaction` • `CommitTransaction` • `EndSession` • `ExecuteStatement` • `FetchPage` • `StartSession` • `StartTransaction` Para saber como QLDB usar esses comandos para gerenciar operações de dados, consulteGerenciamento da sessão com o driver.

Criação de CloudWatch alarmes para monitorar a Amazon QLDB

Você pode criar um CloudWatch alarme da Amazon que envia uma mensagem do Amazon Simple Notification Service (AmazonSNS) quando o alarme muda de estado. Um alarme observa uma única métrica por um período tempo que você especifica. Ele executa uma ou mais ações com base no

valor da métrica em relação a um limite especificado ao longo de vários períodos. A ação é uma notificação enviada para um SNS tópico da Amazon ou uma política de Auto Scaling.

Os alarmes invocam ações somente para mudanças de estado sustentadas. CloudWatch os alarmes não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos.

Para obter mais informações sobre a criação de CloudWatch alarmes, consulte Usando CloudWatch alarmes da Amazon no Guia CloudWatch do usuário da Amazon.

Automatizando a Amazon QLDB com eventos CloudWatch

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon CloudWatch Events permite que você automatize seu Serviços da AWS e responda automaticamente a eventos do sistema, como problemas de disponibilidade de aplicativos ou alterações de recursos. Eventos de Serviços da AWS são entregues aos CloudWatch Eventos quase em tempo real. Você pode escrever regras simples para indicar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. Ações que podem ser automaticamente acionadas incluem:

- Invocando um AWS Lambda função
- Invocando o EC2 comando Amazon Run
- Transmitir o evento Amazon Kinesis Data Streams
- Ativando um AWS Step Functions máquina de estado
- Notificar um SNS tópico da Amazon ou uma fila da Amazon SQS

A Amazon QLDB relata um evento à CloudWatch Events sempre que o estado de um recurso contábil em seu Conta da AWS mudanças. Atualmente, os eventos são emitidos de at-least-once forma garantida, apenas para recursos QLDB contábeis.

A seguir está um exemplo de um evento QLDB relatado, no qual o estado de um livro contábil mudou para`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"
    }
}
```

Alguns exemplos de uso de CloudWatch Eventos com QLDB podem incluir, mas não estão limitados ao seguinte:

- Ativação da função do Lambda sempre que um novo ledger é inicialmente criada `CREATING` no estado e eventualmente se torna `ACTIVE`.
- Notificar um SNS tópico da Amazon quando o estado do seu livro é alterado para `DELETING` e depois para. `DELETED`

Para obter mais informações, consulte o Guia do usuário do Amazon CloudWatch Events.

Registrando QLDB API chamadas da Amazon com AWS CloudTrail

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB está integrada com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um AWS service (Serviço da AWS) emQLDB. CloudTrail captura todas as API chamadas de gerenciamento de recursos QLDB como eventos. As chamadas capturadas incluem chamadas do QLDB console e chamadas de código para as QLDB API operações. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos

para um bucket do Amazon Simple Storage Service (Amazon S3), incluindo eventos para. QLDB Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feitaQLDB, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre CloudTrail, inclusive como configurá-lo e ativá-lo, consulte o AWS CloudTrail Guia do usuário.

QLDBinformações em CloudTrail

CloudTrail está habilitado em seu Conta da AWS quando você cria a conta. Quando a atividade suportada ocorre emQLDB, essa atividade é registrada em um CloudTrail evento junto com outro AWS service (Serviço da AWS) eventos no Histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em seu Conta da AWS. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em seu Conta da AWS, incluindo eventos paraQLDB, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, a trilha se aplica a todos Regiões da AWS. A trilha registra eventos de todas as regiões do AWS particiona e entrega os arquivos de log para o bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros Serviços da AWS para analisar e agir ainda mais com base nos dados do evento coletados nos CloudTrail registros.

Para obter mais informações, consulte os tópicos a seguir no AWS CloudTrail Guia do usuário:

- Visão geral da criação de uma trilha
- CloudTrail serviços e integrações suportados
- Configurando SNS notificações da Amazon para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões
- Recebendo arquivos de CloudTrail log de várias contas

Todas as API ações de gerenciamento de QLDB recursos e dados não transacionais são registradas CloudTrail e documentadas na referência da Amazon. QLDB API Por exemplo, chamadas para as `DeleteLedger` ações `CreateLedgerDescribeLedger`, e geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou do
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado
- Se a solicitação foi feita por outro AWS service (Serviço da AWS)

Para obter mais informações, consulte o CloudTrail userIdentityelemento.

Entendendo as entradas do arquivo de QLDB log

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das API chamadas públicas, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra essas ações:

- `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"
    }
  ]
}
```

Validação de conformidade para a Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Auditores terceirizados avaliam a segurança e a conformidade da Amazon QLDB como parte de vários AWS programas de conformidade, incluindo, mas não se limitando aos seguintes:

- Controles de sistema e organização (SOC)
- Setor de cartões de pagamento (PCI)
- Organização Internacional de Padronização () ISO
- Programa de Gestão e Avaliação da Segurança do Sistema de Informação (ISMAP)
- Lei de Portabilidade e Responsabilidade de Seguros de Saúde () HIPAA

> **Note**
>
> Essa não é uma lista abrangente das QLDB certificações da Amazon.

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS no escopo do Programa de Conformidade e escolha o programa de conformidade no qual você está interessado. Para obter informações gerais, consulte AWS Programas de conformidade .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixando relatórios em AWS Artifact.

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinado pela confidencialidade de seus dados, pelos objetivos de conformidade da sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos em AWS focados em segurança e conformidade.
- Arquitetura para HIPAA segurança e conformidade na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar HIPAA aplicativos elegíveis.

> **Note**
>
> Nem todos Serviços da AWS são HIPAA elegíveis. Para obter mais informações, consulte a Referência de serviços HIPAA elegíveis.

- AWS Recursos de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeie a orientação para controles de segurança em várias estruturas (incluindo Instituto Nacional de Padrões e Tecnologia (NIST), Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e Organização Internacional de Padronização (ISO)).
- Avaliando recursos com regras no AWS Config Guia do desenvolvedor — O AWS Config O serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes do setor e os regulamentos.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança em AWS. O Security Hub usa controles de segurança para avaliar sua AWS recursos e para verificar sua conformidade com os padrões e as melhores práticas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças ao seu Contas da AWS, cargas de trabalho, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, por exemplo PCIDSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A ferramenta AWS a infraestrutura global é construída em torno de Regiões da AWS e zonas de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as Zonas de Disponibilidade, é possível projetar e operar aplicações e bancos de dados que executem o failover automaticamente entre as Zonas de Disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre Regiões da AWS e zonas de disponibilidade, consulte AWS Infraestrutura global.

Durabilidade do armazenamento

QLDBo armazenamento de diários apresenta replicação síncrona para várias zonas de disponibilidade em confirmações de transações. Isso garante que mesmo uma falha total do armazenamento do diário na zona de disponibilidade não comprometeria a integridade dos dados ou a capacidade de manter um serviço ativo. Além disso, o QLDB diário apresenta arquivamentos assíncronos para armazenamento tolerante a falhas. Esse atributo oferece suporte à recuperação de desastres no caso altamente improvável de falha simultânea de armazenamento em várias zonas de disponibilidade.

QLDBo armazenamento indexado é apoiado pela replicação em várias zonas de disponibilidade. Isso garante que mesmo uma falha total do armazenamento indexado na zona de disponibilidade não comprometeria a integridade dos dados ou a capacidade de manter um serviço ativo.

Atributos de durabilidade de dados

Além do AWS infraestrutura global, QLDB oferece os seguintes recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

QLDBcaracterísticas do serviço

Exportação de diário sob demanda

QLDBfornece um recurso de exportação de diário sob demanda. Acesse o conteúdo do seu diário exportando blocos de diário do seu ledger para um bucket do Amazon S3. Você pode usar esses dados para várias finalidades, como retenção de dados, análise e auditoria. Para obter mais informações, consulte Exportação de dados de periódicos da Amazon QLDB.

Backup e restauração

A restauração automatizada para exportações não é suportada no momento. A exportação fornece a capacidade básica de criar redundância adicional de dados na frequência definida. No entanto, um cenário de restauração depende do aplicativo, em que os registros exportados são presumivelmente gravados em um novo ledger usando o mesmo método de ingestão ou similar.

QLDBnão fornece um recurso dedicado de backup e restauração relacionado no momento.

Fluxos de diários

QLDBtambém fornece uma capacidade de fluxo contínuo de diários. Você pode integrar fluxos de QLDB diários com a plataforma de streaming Amazon Kinesis para processar dados de diários em tempo real. Para obter mais informações, consulte Streaming de dados de diários da Amazon QLDB.

QLDBrecurso de design

QLDBfoi projetado para ser resiliente contra corrupção lógica. O QLDB diário é imutável, garantindo que todas as transações confirmadas persistam no diário. Além disso, todas as alterações confirmadas no documento são registradas, pois isso permite a point-in-timevisibilidade de quaisquer alterações não intencionais nos dados contábeis.

QLDBnão fornece um recurso de recuperação automatizada para cenários de corrupção lógica no momento.

Segurança da infraestrutura na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Como um serviço gerenciado, a Amazon QLDB é protegida pelo AWS procedimentos globais de segurança de rede descritos no whitepaper Amazon Web Services: Visão geral dos processos de segurança.

Você usa AWS APIchamadas publicadas para acesso QLDB pela rede. Os clientes devem oferecer suporte ao Transport Layer Security (TLS) 1.0 ou posterior. Recomendamos TLS 1.2 ou posterior. Os clientes também devem oferecer suporte a pacotes de criptografia com sigilo direto perfeito (), como Ephemeral Diffie-Hellman (PFS) ou Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando credenciais programáticas associadas a um IAM diretor. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Você também pode usar um endpoint de nuvem privada virtual (VPC) paraQLDB. VPCOs endpoints de interface permitem que seus VPC recursos da Amazon usem seus endereços IP privados para acessar QLDB sem exposição à Internet pública. Para obter mais informações, consulte Acesse a Amazon QLDB usando um endpoint de interface (AWS PrivateLink).

Acesse a Amazon QLDB usando um endpoint de interface (AWS PrivateLink)

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Você pode usar: AWS PrivateLink para criar uma conexão privada entre você VPC e a AmazonQLDB. Você pode acessar QLDB como se estivesse no seuVPC, sem o uso de um gateway de internet, NAT dispositivo, VPN conexão ou AWS Direct Connect conexão. As instâncias em seu VPC não precisam de endereços IP públicos para serem acessadasQLDB.

Você estabelece essa conexão privada criando um endpoint de interface, alimentado por AWS PrivateLink. Criamos uma interface de rede de endpoint em cada sub-rede que você habilita para o endpoint da interface. Essas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado a. QLDB

Para obter mais informações, consulte Access Serviços da AWS através AWS PrivateLink no AWS PrivateLink Guia.

Tópicos

- Considerações para o QLDB
- Criar um endpoint de interface para QLDB
- Crie uma política de endpoint para seu endpoint de interface.
- Disponibilidade de terminais de interface para QLDB

Considerações para o QLDB

Antes de configurar um endpoint de interface paraQLDB, revise Considerações no AWS PrivateLink Guia.

> **Note**
>
> QLDBsó suporta fazer chamadas para os dados transacionais da QLDBsessão API por meio do endpoint da interface. Isso API inclui somente a SendCommandoperação. No modo de `STANDARD` permissões de um livro contábil, você pode controlar as permissões para ações específicas do PartiQL neste. API

Criar um endpoint de interface para QLDB

Você pode criar um endpoint de interface para QLDB usar o VPC console da Amazon ou o AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criar um endpoint de interface no AWS PrivateLink Guia.

Crie um endpoint de interface para QLDB usar o seguinte nome de serviço:

```
com.amazonaws.region.qldb.session
```

Se você habilitar private DNS para o endpoint da interface, poderá fazer API solicitações QLDB usando seu DNS nome regional padrão. Por exemplo, `session.qldb.us-east-1.amazonaws.com`.

Crie uma política de endpoint para seu endpoint de interface.

Uma política de endpoint é um IAM recurso que você pode anexar a um endpoint de interface. A política de endpoint padrão permite acesso total QLDB por meio do endpoint da interface. Para controlar o acesso permitido pelo seuVPC, anexe uma política QLDB de endpoint personalizada ao endpoint da interface.

Uma política de endpoint especifica as seguintes informações:

- Os diretores que podem realizar ações (Contas da AWS, usuários e funções).
- As ações que podem ser executadas.
- Os recursos nos quais as ações podem ser executadas.

Para obter mais informações, consulte Controlar o acesso aos serviços usando políticas de endpoint no AWS PrivateLink Guia.

Você também pode usar o campo `Condition` em uma política anexada a um usuário, grupo ou função para permitir acesso somente a partir de um endpoint de interface especificado. Quando usadas em conjunto, as políticas e IAM políticas de endpoint podem restringir o acesso a QLDB ações específicas em livros contábeis específicos para um endpoint de interface especificado.

Exemplo de política de endpoint: restringir o acesso a um livro contábil específico QLDB

Veja a seguir um exemplo de uma política de endpoint personalizada paraQLDB. Quando você anexa essa política ao seu endpoint de interface, ela concede acesso à ação `SendCommand` e às ações somente para leitura do PartiQL para todas as entidades principais no recurso contábil especificado. Neste exemplo, o ledger deve estar no modo de permissões `STANDARD`.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`* e *`myExampleLedger`* no exemplo com suas próprias informações.

```
{
   "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"
         ]
      }
   ]
}
```

IAMexemplo de política: restringir o acesso a um QLDB livro contábil somente a partir de um endpoint de interface específico

Veja a seguir um exemplo de uma política IAM baseada em identidade para. QLDB Quando você anexa essa política a um usuário, função ou grupo, ela permite que `SendCommand` acesse um recurso contábil somente a partir do endpoint de interface especificado.

Para usar essa política, substitua *`us-east-1`*, *`123456789012`*, *`myExampleLedger`* e *`vpce-1a2b3c4d`* no exemplo com suas próprias informações.

```
{
   "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"
               }
            }
         }
      ]
}
```

Disponibilidade de terminais de interface para QLDB

A Amazon QLDB oferece suporte a endpoints de interface com políticas em todos os Regiões da AWS onde QLDB está disponível. Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no Referência geral da AWS.

Solução de problemas da Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

As seções a seguir fornecem uma lista agregada de erros comuns que você pode encontrar ao usar a Amazon QLDB e orientações sobre como solucioná-los.

Para obter orientações específicas sobre solução de problemas de IAM acesso, consulteSolução de problemas de QLDB identidade e acesso da Amazon.

Para obter as melhores práticas para ajustar suas instruções partiQL, consulte Otimizar a performance da consulta.

Tópicos

- Executando transações usando o QLDB driver
- Exportação de dados do diário
- Streaming de dados do diário
- Verificação dos dados do diário

Executando transações usando o QLDB driver

Esta seção lista exceções comuns que o QLDB driver da Amazon pode retornar quando você o usa para executar transações partiQL em um livro contábil. Para obter mais informações sobre esse recurso, consulte Conceitos básicos do driver. Para obter as melhores práticas para configurar e usar o driver, consulte Recomendações de driver.

Cada exceção inclui a mensagem de erro específica, seguida por uma breve descrição e sugestões de possíveis soluções.

CapacityExceededException

Mensagem: Capacidade excedida

A Amazon QLDB rejeitou a solicitação porque ela excedeu a capacidade de processamento do livro contábil. QLDBimpõe um limite de escala interno por livro contábil para manter a integridade e o desempenho do serviço. Esse limite varia de acordo com o tamanho da workload de cada solicitação individual. Por exemplo, uma solicitação pode ter uma workload maior se realizar transações de dados ineficientes, como varreduras de tabelas resultantes de uma consulta qualificada não indexada.

Recomendamos que você espere antes de repetir a solicitação. Se seu aplicativo encontrar essa exceção de forma consistente, otimize suas instruções e diminua a taxa e o volume das solicitações que você envia para o ledger. Exemplos de otimização de instruções incluem a execução de menos instruções por transação e o ajuste dos índices da tabela. Para saber como otimizar instruções e evitar varreduras de tabelas, consulte Otimizar a performance da consulta.

Também recomendamos usar a versão mais recente do QLDB driver. O driver tem uma política de repetição padrão que usa recuo exponencial e Jitter para repetir automaticamente exceções como essa. O conceito por detrás do recuo exponencial é usar esperas progressivamente mais longas entre as novas tentativas para respostas de erro consecutivas.

InvalidSessionException

Mensagem: Transação `transactionId` expirou

Uma transação excedeu sua vida útil máxima. Uma transação pode ser executada por até 30 segundos antes de ser confirmada. Após esse limite de tempo limite, qualquer trabalho realizado na transação é rejeitado e a sessão é QLDB descartada. Esse limite protege o cliente de sessões vazadas ao iniciar transações e não confirmá-las ou cancelá-las.

Se essa for uma exceção comum em seu aplicativo, é provável que as transações estejam simplesmente demorando muito para serem executadas. Se o runtime da transação estiver demorando mais de 30 segundos, otimize seus extratos para acelerar as transações. Exemplos de otimização de instruções incluem a execução de menos instruções por transação e o ajuste dos índices da tabela. Para ter mais informações, consulte Otimizar a performance da consulta.

InvalidSessionException

Mensagem: Sessão `sessionId` expirou

QLDBdescartou a sessão porque ela excedeu sua vida útil total máxima. QLDBdescarta as sessões após 13 a 17 minutos, independentemente de uma transação ativa. As sessões podem ser perdidas ou prejudicadas por vários motivos, como falha de hardware, falha de rede ou

reinicialização de aplicativos. Portanto, QLDB impõe uma vida útil máxima às sessões para garantir que o software cliente seja resiliente a falhas de sessão.

Se você encontrar essa exceção, recomendamos que você adquira uma nova sessão e repita a transação. Também recomendamos usar a versão mais recente do QLDB driver, que gerencia o pool de sessões e sua integridade em nome do aplicativo.

InvalidSessionException

Mensagem: Essa sessão não existe

O cliente tentou fazer transações QLDB usando uma sessão que não existe. Supondo que o cliente esteja usando uma sessão que existia anteriormente, a sessão pode não existir mais devido a um dos seguintes motivos:

- Se uma sessão estiver envolvida em uma falha interna do servidor (ou seja, um erro com o código de HTTP resposta 500), QLDB pode optar por descartar a sessão completamente, em vez de permitir que o cliente faça transações com uma sessão de estado incerto. Então, qualquer nova tentativa nessa sessão falhará com esse erro.
- As sessões expiradas acabam sendo esquecidas porQLDB. Então, qualquer tentativa de continuar usando a sessão resultará nesse erro, em vez do `InvalidSessionException` inicial.

Se você encontrar essa exceção, recomendamos que você adquira uma nova sessão e repita a transação. Também recomendamos usar a versão mais recente do QLDB driver, que gerencia o pool de sessões e sua integridade em nome do aplicativo.

RateExceededException

Mensagem: A taxa foi excedida

QLDBlimitou um cliente com base na identidade do chamador. QLDBimpõe a limitação por região e por conta usando um algoritmo de limitação de token bucket. QLDBfaz isso para ajudar no desempenho do serviço e para garantir o uso justo para todos os QLDB clientes. Por exemplo, tentar adquirir um grande número de sessões simultâneas usando a operação `StartSessionRequest` pode levar ao controle de utilização.

Para manter a integridade do aplicativo e reduzir ainda mais o controle de utilização, você pode tentar novamente essa exceção usando recuo exponencial e jitter. O conceito por detrás do recuo exponencial é usar esperas progressivamente mais longas entre as novas tentativas para respostas de erro consecutivas. Recomendamos usar a versão mais recente do QLDB driver.

O driver tem uma política de repetição padrão que usa recuo exponencial e Jitter para repetir automaticamente exceções como essa.

A versão mais recente do QLDB driver também pode ajudar se seu aplicativo estiver constantemente sendo limitado QLDB para `StartSessionRequest` fazer chamadas. O driver mantém um pool de sessões que são reutilizadas em todas as transações, o que pode ajudar a reduzir o número de chamadas `StartSessionRequest` que seu aplicativo faz. Para solicitar um aumento nos limites de API limitação, entre em contato com o AWS Support Centro.

LimitExceededException

Mensagem: Excedeu o limite da sessão

Um ledger excedeu sua cota (também conhecida como limite) no número de sessões ativas. Essa cota é definida em Cotas e limites na Amazon QLDB. Eventualmente, a contagem de sessões ativas de um ledger é consistente, e ledgers consistentemente próximos à cota podem ver essa exceção periodicamente.

Para manter a integridade do seu aplicativo, recomendamos tentar novamente essa exceção. Para evitar essa exceção, certifique-se de não ter configurado mais de 1.500 sessões simultâneas para serem usadas em um único ledger em todos os clientes. Por exemplo, você pode usar o maxConcurrentTransactionsmétodo do QLDBdriver Amazon para Java para configurar o número máximo de sessões disponíveis em uma instância do driver.

QldbClientException

Mensagem: Um resultado transmitido só é válido quando a transação principal está aberta

A transação está fechada e não pode ser usada para recuperar os resultados. QLDB Uma transação é fechada quando é confirmada ou cancelada.

Essa exceção ocorre quando o cliente está trabalhando diretamente com o `Transaction` objeto e está tentando recuperar os resultados QLDB após confirmar ou cancelar uma transação. Para mitigar esse problema, o cliente deve ler os dados antes de fechar a transação.

Exportação de dados do diário

Esta seção lista exceções comuns que QLDB podem retornar quando você exporta dados do diário de um livro contábil para um bucket do Amazon S3. Para obter mais informações sobre esse recurso, consulte Exportação de dados de periódicos da Amazon QLDB.

Cada exceção inclui a mensagem de erro específica, seguida por uma breve descrição e sugestões de possíveis soluções.

AccessDeniedException

Mensagem: Usuário: *`userARN`* não está autorizado a executar: iam: PassRole no recurso: *`roleARN`*

Você não tem permissões para passar uma IAM função para o QLDB serviço. QLDBrequer uma função para todas as solicitações de exportação de periódicos, e você deve ter permissões para passar essa função paraQLDB. A função QLDB fornece permissões de gravação em seu bucket específico do Amazon S3.

Verifique se você define uma IAM política que conceda permissão para realizar a `PassRole` API operação no recurso de IAM função especificado para o QLDB serviço (`qldb.amazonaws.com`). Para ver um exemplo de política, consulte Exemplos de políticas baseadas em identidade para a Amazon QLDB.

IllegalArgumentException

Mensagem: QLDB encontrou um erro ao validar a configuração do S3: *`errorCode errorMessage`*

Uma possível causa desse erro é que o bucket do Amazon S3 fornecido não existe no Amazon S3. Ou QLDB não tem permissões suficientes para gravar objetos no bucket especificado do Amazon S3.

Verifique se o nome do bucket do S3 fornecido na solicitação de trabalho de exportação está correto. Para obter informações sobre a nomenclatura de buckets, consulte Restrições e limitações do bucket no Guia do usuário do Amazon Simple Storage Service.

Além disso, verifique se você define uma política para o bucket especificado que concede `PutObject` e `PutObjectAcl` autoriza o QLDB serviço (`qldb.amazonaws.com`). Para saber mais, consulte Permissões de exportação.

IllegalArgumentException

Mensagem: Resposta inesperada do Amazon S3 ao validar a configuração do S3. Resposta do S3: *`errorCode errorMessage`*

A tentativa de gravar dados de exportação do diário no bucket do S3 fornecido falhou com a resposta de erro fornecida pelo Amazon S3. Para obter mais informações sobre as causas

possíveis, consulte Solução de problemas do Amazon S3 no Guia do usuário do Amazon Simple Storage Service.

IllegalArgumentException

Mensagem: O prefixo do bucket do Amazon S3 não deve exceder 128 caracteres

O prefixo fornecido na solicitação de exportação do diário contém mais de 128 caracteres.

IllegalArgumentException

Mensagem: A data de início não deve ser maior que a data de término

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de data e hora ISO8601 e no Tempo Universal Coordenado ()UTC.

IllegalArgumentException

Mensagem: A data de término não pode ser no futuro

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de `ISO 8601` data e hora e emUTC.

IllegalArgumentException

Mensagem: A configuração de criptografia de objeto fornecida (S3EncryptionConfiguration) não é compatível com uma chave AWS Key Management Service (AWS KMS)

Você forneceu um `KMSKeyArn` com um `ObjectEncryptionType` de `NO_ENCRYPTION` ou `SSE_S3`. Você só pode fornecer um cliente gerenciado AWS KMS key para um tipo de criptografia de objeto de `SSE_KMS`. Para obter mais informações sobre o uso da criptografia no lado do servidor no Amazon S3, consulte Proteger dados usando criptografia no lado do servidor, no Guia do desenvolvedor do Amazon S3.

LimitExceededException

Mensagem: Excedeu o limite de 2 trabalhos de exportação de diário em execução simultânea

QLDBimpõe um limite padrão de dois trabalhos simultâneos de exportação de diário.

Streaming de dados do diário

Esta seção lista exceções comuns que QLDB podem retornar quando você transmite dados do diário de um livro contábil para o Amazon Kinesis Data Streams. Para obter mais informações sobre esse recurso, consulte Streaming de dados de diários da Amazon QLDB.

Cada exceção inclui a mensagem de erro específica, seguida por uma breve descrição e sugestões de possíveis soluções.

AccessDeniedException

Mensagem: Usuário: *`userARN`* não está autorizado a executar: iam: PassRole no recurso: *`roleARN`*

Você não tem permissões para passar uma IAM função para o QLDB serviço. QLDBrequer uma função para todas as solicitações de stream do diário, e você deve ter permissões para transmitir essa função paraQLDB. A função QLDB fornece permissões de gravação em seu recurso específico do Amazon Kinesis Data Streams.

Verifique se você define uma IAM política que conceda permissão para realizar a `PassRole` API operação no recurso de IAM função especificado para o QLDB serviço (`qldb.amazonaws.com`). Para ver um exemplo de política, consulte Exemplos de políticas baseadas em identidade para a Amazon QLDB.

IllegalArgumentException

Mensagem: QLDB encontrou um erro ao validar o Kinesis Data Streams: Resposta do Kinesis: *`errorCode errorMessage`*

Uma possível causa desse erro é que o recurso Kinesis Data Streams fornecido não existe. Ou QLDB não tem permissões suficientes para gravar registros de dados no stream de dados especificado do Kinesis.

Verifique se o fluxo de dados do Kinesis que você fornece na sua solicitação de fluxo está correto. Para obter mais informações, consulte Criar e atualizar fluxos de dados no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Além disso, verifique se você define uma política para o stream de dados do Kinesis especificado que conceda ao QLDB service (`qldb.amazonaws.com`) permissões para as seguintes ações. Para ter mais informações, consulte Permissões de fluxo.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Mensagem: Resposta inesperada do Kinesis Data Streams ao validar a configuração do Kinesis. Resposta do Kinesis: *`errorCode errorMessage`*

A tentativa de gravar registros de dados no fluxo de dados do Kinesis fornecido falhou com a resposta de erro fornecida do Kinesis. Para obter mais informações sobre as causas possíveis, consulte Solução de problemas dos produtores do Amazon Kinesis Data Streams no Guia do usuário do Amazon Kinesis Data Streams.

IllegalArgumentException

Mensagem: A data de início não deve ser maior que a data de término.

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de data e hora ISO8601 e no Tempo Universal Coordenado ()UTC.

IllegalArgumentException

Mensagem: A data de início não pode ser no futuro

Ambos `InclusiveStartTime` `ExclusiveEndTime` devem estar no formato de `ISO 8601` data e hora e emUTC.

LimitExceededException

Mensagem: Excedeu o limite de 5 fluxos de registro em execução simultânea no Kinesis Data Streams

QLDBimpõe um limite padrão de cinco fluxos de diário simultâneos.

Verificação dos dados do diário

Esta seção lista as exceções comuns que QLDB podem retornar quando você verifica os dados do diário em um livro contábil. Para obter mais informações sobre esse recurso, consulte Verificação de dados na Amazon QLDB.

Cada exceção inclui a mensagem de erro específica, seguida pelas API operações que podem gerá-la, uma breve descrição e sugestões de possíveis soluções.

IllegalArgumentException

Mensagem: O valor de Ion fornecido não é válido e não pode ser analisado.

APIoperações: `GetDigest, GetBlock, GetRevision`

Certifique-se de fornecer um valor válido do Amazon Ion antes de tentar novamente sua solicitação.

IllegalArgumentException

Mensagem: O endereço de bloco fornecido não é válido.

APIoperações: `GetDigest, GetBlock, GetRevision`

Certifique-se de fornecer um valor válido de endereço de bloco antes de tentar novamente sua solicitação. Um endereço de bloco é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`

IllegalArgumentException

Mensagem: O número de sequência do endereço da dica de resumo fornecido está além do último registro confirmado da cadeia.

APIoperações: `GetDigest, GetBlock, GetRevision`

O endereço da dica de resumo que você fornecer deve ter um número de sequência menor ou igual ao número de sequência do último registro confirmado da vertente do diário. Antes de repetir sua solicitação, certifique-se de fornecer um endereço de sugestão com um número de sequência válido.

IllegalArgumentException

Mensagem: O ID de cadeia do endereço de bloco fornecido não é válido.

APIoperações: `GetDigest, GetBlock, GetRevision`

O endereço de bloco que você fornece deve ter um ID de cadeia que corresponda ao ID da cadeia do diário. Certifique-se de fornecer um valor válido de endereço de bloco antes de tentar novamente com um ID de cadeia.

IllegalArgumentException

Mensagem: O número de sequência do endereço de bloco fornecido está além do último registro confirmado da cadeia.

APIoperações: `GetBlock, GetRevision`

O endereço de bloco que você fornece deve ter um número de sequência menor ou igual ao número de sequência do último registro confirmado da cadeia. Antes de repetir sua solicitação, certifique-se de fornecer um endereço de bloco com um número de sequência válido.

IllegalArgumentException

Mensagem: O ID de cadeia do endereço de bloco fornecido deve corresponder ao ID de cadeia do endereço da dica de resumo fornecido.

APIoperações: `GetBlock, GetRevision`

Você só pode verificar uma revisão ou bloco de documento se ele existir na mesma linha do diário que o resumo fornecido.

IllegalArgumentException

Mensagem: O número de sequência do endereço de bloco fornecido não deve ser maior que o número de sequência do endereço de dica de resumo fornecido.

APIoperações: `GetBlock, GetRevision`

Você só pode verificar uma revisão ou bloco de documento se estiver coberto pelo mesmo resumo que você forneceu. Isso significa que foi entregue ao diário antes do endereço da dica de resumo.

IllegalArgumentException

Mensagem: O ID do documento fornecido não foi encontrado no bloco no endereço de bloco especificado.

APIoperação: `GetRevision`

O ID do documento que você fornece deve existir no endereço de bloco fornecido. Antes de repetir sua solicitação, verifique se esses dois parâmetros são consistentes.

Referência do Amazon QLDB PartiQL

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB oferece suporte a um subconjunto da linguagem de consulta partiQL. Os tópicos a seguir descrevem a QLDB implementação do partiQL.

> **Note**
>
> - QLDBnão oferece suporte a todas as operações partiQL.
> - Todas as instruções partiQL em QLDB estão sujeitas aos limites de transação, conforme definido em. Cotas e limites na Amazon QLDB
> - Esta referência fornece exemplos básicos de sintaxe e uso de instruções partiQL que você executa manualmente no console ou QLDB QLDB no shell. Para exemplos de código que mostram como executar programaticamente instruções semelhantes usando o QLDB driver, consulte os tutoriais em. Conceitos básicos do driver

Tópicos

- O que é PartiQL?
- PartiQL na Amazon QLDB
- Dicas rápidas do partiQL em QLDB
- Convenções de referência do Amazon QLDB PartiQL
- Tipos de dados na Amazon QLDB
- QLDBDocumentos da Amazon
- Consultando o Ion com o PartiQL na Amazon QLDB
- Comandos partiQL na Amazon QLDB
- Funções partiQL na Amazon QLDB

- Procedimentos armazenados em partiQL na Amazon QLDB
- Operadores partiQL na Amazon QLDB
- Palavras-chave reservadas na Amazon QLDB
- Referência de formato de dados Amazon Ion na Amazon QLDB

O que é PartiQL?

O PartiQL fornece acesso a consultas SQL compatível em vários armazenamentos de dados contendo dados estruturados, dados semiestruturados e dados aninhados. É amplamente usado na Amazon e agora está disponível como parte de muitos Serviços da AWS, inclusiveQLDB.

Para obter a especificação da PartiQL e um tutorial sobre a linguagem de consulta principal, consulte a Documentação da PartiQL.

O partiQL estende SQL-92 para suportar documentos no formato de dados Amazon Ion. Para obter mais informações sobre o Amazon Ion, consulte o Referência de formato de dados Amazon Ion na Amazon QLDB.

PartiQL na Amazon QLDB

Para executar consultas PartiQL noQLDB, você pode usar uma das seguintes opções:

- O editor partiQL no for AWS Management Console QLDB
- O QLDB shell da linha de comando
- Um QLDB driver AWS fornecido para executar consultas programaticamente

Para obter informações sobre como usar esses métodos para acessarQLDB, consulteAcessando a Amazon QLDB.

Para saber como controlar o acesso para executar cada comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Dicas rápidas do partiQL em QLDB

A seguir está um breve resumo das dicas e das melhores práticas para trabalhar com o PartiQL em: QLDB

- Entenda os limites de simultaneidade e transação — Todas as declarações, incluindo `SELECT` consultas, estão sujeitas a conflitos otimistas de controle de simultaneidade (OCC) e limites de transação, incluindo um tempo limite de transação de 30 segundos.
- Use índices: use índices de alta cardinalidade e execute consultas direcionadas para otimizar suas declarações e evitar a varredura completa da tabela. Para saber mais, consulte Otimizar a performance da consulta.
- Use predicados de igualdade: as pesquisas indexadas exigem um operador de igualdade (= ou `IN`). Operadores de desigualdade (<, >, `LIKE`, `BETWEEN`) não se qualificam para pesquisas indexadas e resultam em verificações de tabela completa.
- Use somente junções internas — QLDB suporta somente junções internas. Como prática recomendada, junte em campos indexados para cada tabela que você está unindo. Escolha índices de alta cardinalidade para os critérios de junção e os predicados de igualdade.

Convenções de referência do Amazon QLDB PartiQL

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção explica as convenções usadas para escrever a sintaxe dos comandos, funções e expressões do PartiQL descritos na referência do Amazon PartiQL. QLDB Essas convenções não devem ser confundidas com a sintaxe e a semântica da própria linguagem de consulta partiQL.

Caractere	Descrição
CAPS	Palavras em letras maiúsculas são palavras-chave.
[ ]	Parênteses denotam cláusulas ou argumentos opcionais. Vários argumento s em parênteses indicam que você pode escolher qualquer número de argumentos. Além disso, argumentos entre colchetes em linhas separadas indicam que o QLDB analisador espera que os argumentos estejam na ordem em que estão listados na sintaxe.

Caractere	Descrição
|	Barras verticais indicam que você pode escolher entre os argumentos.
red italics	As palavras em vermelho itálico indicam espaços reservados. Insira o valor apropriado no lugar da palavra em vermelho itálico.
. . .	Uma elipse indica que você pode repetir o elemento anterior.
'	Valores entre aspas simples indicam que você deve digitar as aspas. No PartiQL, aspas simples denotam valores de string ou nomes de campos nas estruturas Amazon Ion.
"	Valores entre aspas duplas indicam que você deve digitar as aspas duplas. No PartiQL, aspas duplas denotam identificadores entre aspas.
`	Os valores em acentos graves indicam que você deve inserir os acentos graves. No PartiQL, acentos graves denotam valores literais de Ion.

Tipos de dados na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB armazena documentos no formato Amazon Ion. O Amazon Ion é um formato de serialização de dados (tanto em formato de texto quanto em formato de codificação binária) que é um superconjunto de. JSON A tabela a seguir lista os tipos de dados de íons que você pode usar em QLDB documentos.

Tipo de dados	Descrição
`null`	Um valor nulo genérico

Tipo de dados	Descrição
`bool`	Valores boolianos
`int`	Números inteiros assinados de tamanho arbitrário
`decimal`	Números reais codificados em decimal de precisão arbitrária
`float`	Números de ponto flutuante codificados em binário (64 bits) IEEE
`timestamp`	Momentos de data/hora/fuso horário de precisão arbitrária
`string`	Literais de texto em Unicode
`symbol`	Átomos simbólicos Unicode (identificadores)
`blob`	Dados binários de codificação definida pelo usuário
`clob`	Dados de texto de codificação definida pelo usuário
`struct`	Coleções não ordenadas de pares nome-valor
`list`	Coleções heterogêneas ordenadas de valores

Consulte o documento de especificação do Ion no GitHub site da Amazon para obter uma lista completa dos tipos de dados principais do Ion com descrições completas e detalhes de formatação de valores.

QLDBDocumentos da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB armazena registros de dados como documentos, que são apenas `struct` objetos Amazon Ion inseridos em uma tabela. Para obter a especificação do Ion, consulte o GitHub site da Amazon Ion.

Tópicos

- Estrutura de documento Ion
- Mapeamento do tipo de PartiQL-Ion
- ID do documento

Estrutura de documento Ion

Por exemploJSON, QLDB os documentos são compostos por pares de nome-valor na estrutura a seguir.

```
{
  name1: value1,
  name2: value2,
  name3: value3,
  ...
  nameN: valueN
}
```

O nome é um token símbolo e os valores são irrestritos. Cada par nome-valor é chamado de campo. O valor de um campo pode ser qualquer um dos Tipos de dados de Ion, incluindo tipos de contêiner: estruturas aninhadas, listas e listas de estruturas.

Da mesma formaJSON, a `struct` é indicado por colchetes (`{...}`) e a `list` é indicado por colchetes (). `[...]` O exemplo a seguir é um documento dos dados de exemplo Começando a usar o QLDB console da Amazon que contém valores de vários tipos.

```
{
    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**
>
> No Ion, aspas duplas denotam valores de string, e símbolos sem aspas representam nomes de campo. Porém, no PartiQL, aspas simples denotam tanto os strings quanto os nomes dos campos.
>
> Essa diferença na sintaxe permite que a linguagem de consulta PartiQL SQL mantenha a compatibilidade e o formato de dados Amazon Ion mantenha a JSON compatibilidade. Para obter detalhes sobre a sintaxe e a semântica do partiQL em, consulte. QLDB Consultar o Ion com o PartiQL

Mapeamento do tipo de PartiQL-Ion

EmQLDB, o PartiQL estende o sistema SQL de tipos do Ion para cobrir o modelo de dados Ion. Esse mapeamento é descrito da seguinte forma:

- SQLos tipos escalares são cobertos por seus equivalentes de íons. Por exemplo: .
  - `CHAR` e `VARCHAR` são sequências Unicode mapeadas para o tipo de íon `string`.
  - `NUMBER` mapeia para o tipo `decimal` de Ion.
- O `struct` tipo de Ion é equivalente a uma SQL tupla, que tradicionalmente representa uma linha da tabela.
  - No entanto, com conteúdo aberto e sem esquema, as consultas que dependem da natureza ordenada de uma SQL tupla não são suportadas (como a ordem de saída de). `SELECT *`

- Além de `NULL`, o partiQL tem um tipo `MISSING`. Essa é uma especialização `NULL` e indica a falta de um campo. Esse tipo é necessário porque os campos `struct` de Ion podem ser esparsos.

ID do documento

QLDBatribui uma ID de documento a cada documento que você insere em uma tabela. Todos os sistemas atribuídos IDs são identificadores universalmente exclusivos (UUID), cada um representado em uma string codificada em Base62 (por exemplo,). `3Qv67yjXEwB9SjmvkuG6Cp` Para ter mais informações, consulte Único IDs na Amazon QLDB.

Cada revisão de documento é identificada exclusivamente por uma combinação do ID do documento e um número de versão baseado em zero.

O ID do documento e os campos de versões estão incluídos nos metadados do documento, que você pode consultar na visualização confirmada (a visualização definida pelo sistema de uma tabela). Para obter mais informações sobre exibições emQLDB, consulteConceitos principais. Para saber mais sobre metadados, consulte Consultar metadados do documento.

Consultando o Ion com o PartiQL na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Ao consultar dados na AmazonQLDB, você escreve declarações no formato partiQL, mas QLDB retorna os resultados no formato Amazon Ion. O partiQL foi projetado para ser SQL compatível, enquanto o Ion é uma extensão do. JSON Isso leva a diferenças sintáticas entre a forma como você anota os dados em suas instruções da consulta e a forma como os resultados da consulta são exibidos.

Esta seção descreve a sintaxe e a semântica básicas para executar instruções partiQL manualmente usando o QLDBconsole ou o shell. QLDB

> **Tip**
>
> Quando você executa consultas partiQL programaticamente, a prática recomendada é usar instruções parametrizadas. Você pode usar um ponto de interrogação (?) como um espaço reservado para a variável de associação em suas instruções para evitar essas regras de sintaxe. Isso também é mais seguro e eficiente.
> Para saber mais, consulte os seguintes tutoriais em Conceitos básicos do driver:
>
> - Java: Tutorial de início rápido | Referência de Cookbook
> - .NET: Tutorial de início rápido | Referência de Cookbook
> - Go: Tutorial de início rápido | Referência de Cookbook
> - Node.js: Tutorial de início rápido | Referência de Cookbook
> - Python: Tutorial de início rápido | Referência de Cookbook

Tópicos

- Sintaxe e semântica
- Notação de acento grave
- Navegação por caminhos
- Aliases
- Especificação PartiQL

Sintaxe e semântica

Ao usar o QLDB console ou o QLDB shell para consultar dados do Ion, veja a seguir a sintaxe e a semântica fundamentais do partiQL:

Diferenciação de letras maiúsculas e minúsculas

Todos os nomes de objetos QLDB do sistema, incluindo nomes de campos, nomes de tabelas e nomes de livros contábeis, diferenciam maiúsculas de minúsculas.

Valores de string

Em Ion, aspas duplas (`"..."`) denotam uma string.

No PartiQL, aspas simples (`'...'`) denotam uma string.

Símbolos e identificadores

Em Ion, aspas simples (`'...'`) denotam um símbolo. Um subconjunto de símbolos em Ion chamados identificadores é representado por texto sem aspas.

Em PartiQL, aspas duplas (`"..."`) denotam identificadores entre aspas, como uma palavra reservada usada como nome de tabela. O texto sem aspas representa um identificador PartiQL regular, como um nome de tabela que não é uma palavra reservada.

Literais de Ion

Qualquer literal de Ion pode ser indicado com acentos graves (`` `...` ``) em uma instrução partiQL.

Nomes de campos

Todos os nomes de campos são símbolos que fazem distinção entre maiúsculas e minúsculas O partiQL permite indicar nomes de campo com aspas simples em uma declaração. DML Essa é uma alternativa abreviada ao uso da função `cast` de PartiQL para definir um símbolo. Também é mais intuitivo do que usar acentos graves para indicar um símbolo literal de Ion.

Literais

Os literais da linguagem de consulta partiQL correspondem aos tipos de dados Ion, da seguinte forma:

Escalares

Siga a SQL sintaxe quando aplicável, conforme descrito na Mapeamento do tipo de PartiQL-Ion seção. Por exemplo: .

- `5`
- `'foo'`
- `null`

Structs

Também conhecidos como tuplos ou objetos em vários formatos e outros modelos de dados.

Indicado por chaves (`{...}`) com `struct` elementos separados por vírgulas.

- `{ 'id' : 3, 'arr': [1, 2] }`

Listas

Também conhecidas como matrizes.

Indicada por colchetes (`[...]`) com elementos separados por vírgulas.

- `[ 1, 'foo' ]`

Bolsas

Coleções não ordenadas no PartiQL.

Indicada por maior e menor duplos (`<<...>>`) com elementos separados por vírgulas. DentroQLDB, uma mesa pode ser considerada uma bolsa. No entanto, uma sacola não pode ser aninhada em documentos em uma tabela.

- `<< 1, 'foo' >>`

Exemplo

Veja a seguir um exemplo da sintaxe de uma instrução `INSERT` com vários tipos de Ion.

```
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`
}
```

Notação de acento grave

O partiQL abrange totalmente todos os tipos de dados Ion, para que você possa gravar qualquer instrução sem usar acentos graves. Porém, há casos em que essa sintaxe literal do Ion pode tornar suas instruções mais claras e concisas.

Por exemplo, para inserir um documento com valores de timestamp e símbolo Ion, você pode escrever a seguinte instrução usando somente a sintaxe partiQL.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

Isso é bastante detalhado, então, em vez disso, você pode usar acentos graves para simplificar sua instrução.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

Você também pode colocar toda a estrutura em acentos graves para economizar mais algumas teclas.

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

Important

Strings e símbolos são classes diferentes no PartiQL. Isso significa que, mesmo que tenham o mesmo texto, não são iguais. Por exemplo, as seguintes expressões PartiQL são avaliadas com valores de Ion diferentes.

```
'foo'
```

```
`foo`
```

Navegação por caminhos

Ao escrever linguagem de manipulação de dados (DML) ou instruções de consulta, você pode acessar campos em estruturas aninhadas usando etapas de caminho. O PartiQL suporta anotação de pontos para acessas os nomes dos campos de uma estrutura principal. O exemplo a seguir acessa o campo `Model` de um `Vehicle` principal.

```
Vehicle.Model
```

Para acessar um elemento específico de uma lista, você pode usar o operador de colchetes para indicar um número ordinal baseado em zero. O exemplo a seguir acessa o elemento de `SecondaryOwners` com um número ordinal de `2`. Em outras palavras, esse é o terceiro elemento da lista.

```
SecondaryOwners[2]
```

Aliases

QLDBoferece suporte a conteúdo e esquema abertos. Portanto, quando você está acessando campos específicos em uma instrução, a melhor maneira de garantir que você obtenha os resultados esperados é usar aliases. Por exemplo, se você não especificar um alias explícito, o sistema gera um implícito para suas fontes `FROM`.

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

Mas os resultados são imprevisíveis para conflitos de nomes de campos. Se outro campo chamado `VIN` existir em uma estrutura aninhada nos documentos, os valores `VIN` retornados por essa consulta podem surpreendê-lo. Como prática recomendada, escreva a seguinte instrução em vez disso. Essa consulta declara `v` como um alias que varia sobre a tabela `Vehicle`. A palavra-chave `AS` é opcional.

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

O uso de aliases é particularmente útil ao criar caminhos em coleções aninhadas em um documento. Por exemplo, a instrução a seguir declara o como um alias que abrange toda a coleção `VehicleRegistration.Owners`.

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

O caractere `@` é tecnicamente opcional aqui. Mas isso indica explicitamente que você deseja a estrutura `Owners` em `VehicleRegistration`, não uma coleção com um nome `Owners` diferente (se houver).

Especificação PartiQL

Para obter mais informações sobre a linguagem de consulta PartiQL, consulte a Especificação PartiQL.

Comandos partiQL na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O partiQL estende SQL -92 para suportar documentos no formato de dados Amazon Ion. A Amazon QLDB oferece suporte aos seguintes comandos partiQL.

Para saber como controlar o acesso para executar cada comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

> **Note**
>
> - QLDBnão oferece suporte a todos os comandos partiQL.
> - Todas as instruções partiQL em QLDB estão sujeitas aos limites de transação, conforme definido em. Cotas e limites na Amazon QLDB
> - Esta referência fornece exemplos básicos de sintaxe e uso de instruções partiQL que você executa manualmente no console ou QLDB QLDB no shell. Para exemplos de código que mostram como executar instruções semelhantes usando uma linguagem de programação suportada, consulte os tutoriais em Conceitos básicos do driver.

DDLdeclarações (linguagem de definição de dados)

A linguagem de definição de dados (DDL) é o conjunto de instruções partiQL que você usa para gerenciar objetos de banco de dados, como tabelas e índices. Você usa DDL para criar e soltar esses objetos.

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- DROP TABLE
- UNDROP TABLE

DMLdeclarações (linguagem de manipulação de dados)

A linguagem de manipulação de dados (DML) é o conjunto de instruções partiQL que você usa para gerenciar dados QLDB em tabelas. Você usa DML instruções para adicionar, modificar ou excluir dados em uma tabela.

As instruções a seguir DML e a linguagem de consulta são suportadas:

- DELETE
- FROM(INSERT,REMOVE, ouSET)
- INSERT
- SELECT
- UPDATE

CREATEINDEXcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `CREATE INDEX` comando para criar um índice para um campo de documento em uma tabela.

Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

> **Important**
>
> QLDBrequer um índice para pesquisar um documento com eficiência. Sem um índice, QLDB precisa fazer uma digitalização completa da tabela ao ler documentos. Isso pode causar problemas de desempenho em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
>
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade (= ou `IN`) em um campo indexado ou em um ID de documento. Para ter mais informações, consulte Otimizar a performance da consulta.

Observe as seguintes restrições ao criar índices:

- Um índice só pode ser criado em um único campo de nível superior. Não há suporte para índices compostos, aninhados, exclusivos e baseados em funções.
- Você pode criar um índice em qualquer tipo de dados Ion, incluindo `list` e `struct`. No entanto, você só pode fazer a pesquisa indexada pela igualdade de todo o valor de Ion, independentemente do tipo de Ion. Por exemplo, ao usar um tipo `list` como índice, você não pode fazer uma pesquisa indexada por um item dentro da lista.
- O desempenho da consulta é aprimorado somente quando você usa um predicado de igualdade; por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`.

  QLDBnão respeita as desigualdades nos predicados de consulta. Como resultado, as verificações filtradas por intervalo não são implementadas.
- Os nomes dos campos indexados diferenciam maiúsculas e minúsculas e podem ter no máximo 128 caracteres.
- A criação do índice em QLDB é assíncrona. O tempo necessário para concluir a criação de um índice em uma tabela não vazia varia dependendo do tamanho da tabela. Para ter mais informações, consulte Gerenciamento de Índices.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno
- Exemplos
- Executar programaticamente usando o driver

Sintaxe

```
CREATE INDEX ON table_name (field)
```

Parâmetros

table_name

O nome da tabela na qual você deseja qual criar o índice. A tabela já deve existir.

O nome da tabela diferencia maiúsculas de minúsculas.

field

O nome do campo do documento para o qual criar o índice. O campo deve ser um atributo de nível superior.

Os nomes dos campos indexados diferenciam maiúsculas e minúsculas e podem ter no máximo 128 caracteres.

Você pode criar um índice em qualquer tipo de dados Amazon Ion, incluindo `list` e `struct`. No entanto, você só pode fazer a pesquisa indexada pela igualdade de todo o valor de Ion, independentemente do tipo de Ion. Por exemplo, ao usar um tipo `list` como índice, você não pode fazer uma pesquisa indexada por um item dentro da lista.

Valor de retorno

`tableId`: o ID exclusivo da tabela na qual você criou o índice.

Exemplos

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

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

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook
- Python: Tutorial de início rápido | Referência de Cookbook

CREATETABLEcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `CREATE TABLE` comando para criar uma nova tabela.

As tabelas têm nomes simples e nenhum namespace. QLDBoferece suporte a conteúdo aberto e não impõe esquema, portanto, você não define atributos ou tipos de dados ao criar tabelas.

> **Note**
>
> Para aprender a controlar o acesso para executar cada comando do PartiQL em um ledger, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros

- Valor de retorno
- Marcar tabelas na criação
- Exemplos
- Executar programaticamente usando o driver

Sintaxe

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

Parâmetros

table_name

O nome exclusivo da tabela a ser criada. Não deve existir já uma tabela ativa com o mesmo nome. As seguintes são nomeadas usando restrições de nomenclatura:

- As tags devem conter apenas 1-128 caracteres alfanuméricos ou sublinhados.
- Deve ter uma letra ou um sublinhado para o primeiro caractere.
- Pode ter qualquer combinação de caracteres alfanuméricos e sublinhados para os caracteres restantes.
- Diferencia maiúsculas de minúsculas.
- Não deve ser uma palavra reservada do QLDB PartiQL.

'***key***': '***value***'

(Opcional) As tags a serem anexadas ao recurso de tabela durante a criação. Cada tag é definida como um par de valores-chave, em que a chave e o valor são indicados por aspas simples. Cada par de valores-chave é definido dentro de uma estrutura Amazon Ion que é indicada por acentos graves.

Atualmente, a marcação de tabelas na criação é suportada apenas para ledgers no modo de permissões *STANDARD*.

Valor de retorno

`tableId`: o ID exclusivo da tabela que você criou.

Marcar tabelas na criação

> **Note**
>
> Atualmente, a marcação de tabelas na criação é suportada apenas para ledgers no modo de permissões `STANDARD`.

Opcionalmente, você pode marcar seus recursos de tabela especificando tags em uma instrução `CREATE TABLE`. Para obter mais informações sobre tags, consulte Marcação de recursos da Amazon QLDB. O exemplo a seguir cria uma tabela chamada `Vehicle` com a tag `environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Marcar tabelas na criação requer acesso às ações `qldb:PartiQLCreateTable` e `qldb:TagResource`. Para saber mais sobre permissões para QLDB recursos, consulteComo a Amazon QLDB trabalha com IAM.

Ao marcar os recursos no momento da criação, você elimina a necessidade de executar scripts personalizados de marcação após a criação do recurso. Depois que uma tabela é marcada, você pode controlar o acesso à tabela com base nessas tags. Por exemplo, você pode conceder acesso total somente a tabelas que tenham uma tag específica. Para obter um exemplo JSON de política, consulteAcesso total a todas as ações com base nas tags da tabela.

Exemplos

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook
- Python: Tutorial de início rápido | Referência de Cookbook

DELETEcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `DELETE` comando para marcar um documento ativo como excluído em uma tabela criando uma revisão nova, mas final, do documento. Essa revisão final indica que o documento foi excluído. Essa operação encerra o ciclo de vida de um documento, o que significa que nenhuma outra revisão do documento com o mesmo ID do documento pode ser criada.

Essa operação é irreversível. Você ainda pode consultar o histórico de revisão de um documento excluído usando Função de histórico.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno
- Exemplos
- Executar programaticamente usando o driver

Sintaxe

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

Parâmetros

table_name

O nome do usuário da tabela que contém o item a ser excluído. DMLas declarações só são suportadas na visualização padrão do usuário. Cada instrução só pode ser executada em uma única tabela.

AS ***table_alias***

(Opcional) Um alias definido pelo usuário que varia em uma tabela a ser excluída. A palavra-chave `AS` é opcional.

BY ***id_alias***

(Opcional) Um alias definido pelo usuário que se vincula ao campo de metadados `id` de cada documento no conjunto de resultados. O alias deve ser declarado na cláusula `FROM` usando a palavra-chave `BY`. Isso é útil quando você deseja filtrar o ID do documento ao consultar a visualização padrão do usuário. Para ter mais informações, consulte Usar a cláusula BY para consultar o ID do documento.

WHERE ***condition***

Os critérios de seleção para os documentos a serem excluídos.

> **Note**
>
> Se você omitir a cláusula `WHERE`, todos os documentos na tabela serão excluídos.

Valor de retorno

`documentId`: o ID exclusivo de cada documento que você excluiu.

Exemplos

```
DELETE FROM VehicleRegistration AS r
```

```
WHERE r.VIN = '1HVBBAANXWH544237'
```

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook
- Python: Tutorial de início rápido | Referência de Cookbook

DROPINDEXcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `DROP INDEX` comando para excluir um índice em uma tabela.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno
- Exemplos

Sintaxe

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> A cláusula `WITH (purge = true)` é obrigatória para todas as instruções `DROP INDEX` e atualmente `true` é o único valor suportado.
> A palavra-chave `purge` diferencia maiúsculas e minúsculas e deve ser toda em minúsculas.

Parâmetros

"***indexId***"

O ID exclusivo do índice a ser removido, indicado por aspas duplas.

ON ***table_name***

O nome da tabela cujo índice você deseja descartar.

Valor de retorno

`tableId`: o ID exclusivo da tabela cujo índice você descartou.

Exemplos

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

DROPTABLEcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `DROP TABLE` comando para desativar uma tabela existente. Você pode usar a instrução UNDROP TABLE para reativá-la. Desativar ou reativar uma tabela não afeta seus documentos ou índices.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno
- Exemplos

Sintaxe

```
DROP TABLE table_name
```

Parâmetros

table_name

O nome da tabela a ser desativada. A tabela já deve existir e ter um status de `ACTIVE`.

Valor de retorno

`tableId`: o ID exclusivo da tabela que você desativou.

Exemplos

```
DROP TABLE VehicleRegistration
```

FROMcomando (INSERTREMOVE,, ouSET) na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, uma declaração que começa com `FROM` é uma extensão PartiQL que permite inserir e remover elementos específicos em um documento. Você também pode usar essa instrução para atualizar elementos existentes em um documento, semelhante ao comando UPDATE.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Coleções aninhadas
- Valor de retorno
- Exemplos
- Executar programaticamente usando o driver

Sintaxe

FROM-INSERT

Insira um novo elemento em um documento existente. Para inserir um novo documento de nível superior em uma tabela, você deve usar INSERT.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

```
INSERT INTO element VALUE data [ AT key_name ]
```

FROM-REMOVE

Remova um elemento existente em um documento ou remova um documento inteiro de nível superior. A última é semanticamente igual à sintaxe DELETE tradicional.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

FROM-SET

Atualize um ou mais elementos em um documento. Se um elemento não existir, ele será inserido. Ela é semanticamente igual à sintaxe UPDATE tradicional.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

Parâmetros

table_name

O nome do usuário da tabela que contém os dados a serem modificados. DMLas declarações só são suportadas na visualização padrão do usuário. Cada instrução só pode ser executada em uma única tabela.

Nessa cláusula, você também pode incluir uma ou mais coleções aninhadas na tabela especificada. Para obter mais detalhes, consulte Coleções aninhadas.

AS ***table_alias***

(Opcional) Um alias definido pelo usuário que varia em uma tabela a ser modificada. Todos os aliases de tabela usados na cláusula `SET`, `REMOVE`, `INSERT INTO` ou `WHERE` devem ser declarados na cláusula `FROM`. A palavra-chave `AS` é opcional.

BY ***id_alias***

(Opcional) Um alias definido pelo usuário que se vincula ao campo de metadados `id` de cada documento no conjunto de resultados. O alias deve ser declarado na cláusula `FROM` usando

a palavra-chave BY. Isso é útil quando você deseja filtrar o ID do documento ao consultar a visualização padrão do usuário. Para ter mais informações, consulte Usar a cláusula BY para consultar o ID do documento.

WHERE ***condition***

Os critérios de seleção para os documentos a serem modificados.

> **Note**
>
> Se você omitir a cláusula WHERE, todos os documentos na tabela serão modificados.

element

Um elemento de documento a ser criado ou modificado.

data

Um novo valor para o elemento.

AT ***key_name***

Um nome da chave a ser adicionado aos documentos a serem modificados. Você deve especificar o VALUE correspondente junto com o nome da chave. Isso é necessário para inserir um novo valor AT em uma posição específica em um documento.

Coleções aninhadas

Embora você possa executar uma DML instrução somente em uma única tabela, você pode especificar coleções aninhadas nos documentos dessa tabela como fontes adicionais. Cada alias que você declara para uma coleção aninhada pode ser usado na cláusula WHERE, SET, INSERT INTO ou cláusula REMOVE.

Por exemplo, as origens FROM da instrução a seguir incluem a tabela VehicleRegistration e a estrutura aninhada Owners.SecondaryOwners.

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

Este exemplo atualiza o elemento específico da lista `SecondaryOwners` que tem um `PersonId` de `'abc123'` dentro do documento `VehicleRegistration` que tem um `VIN` de `'1N4AL11D75C109151'`. Essa expressão permite especificar um elemento de uma lista por seu valor em vez de seu índice.

Valor de retorno

`documentId`: o ID exclusivo de cada documento que você atualizou ou excluiu.

Exemplos

Modifica um elemento em um documento. Se o elemento não existir, ele será inserido.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

Modifica ou insere um elemento e um filtro no campo de metadados `id` do documento atribuído pelo sistema.

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

Modifica o campo `PersonId` do primeiro elemento na lista `Owners.SecondaryOwners` em um documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

Remove um elemento existente em um documento.

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

Remove um documento inteiro de uma tabela.

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

Remove o primeiro elemento da lista `Owners.SecondaryOwners` em um documento na tabela `VehicleRegistration`.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

Insere `{'Mileage':26500}` como um par nome-valor de nível superior em um documento na tabela `Vehicle`.

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

Anexa `{'PersonId':'abc123'}` como um par nome-valor no campo `Owners.SecondaryOwners` de um documento na tabela `VehicleRegistration`. Observe que `Owners.SecondaryOwners` já deve existir e ser um tipo de dados da lista para que essa instrução seja válida. Caso contrário, a palavra-chave `AT` é exigida na cláusula `INSERT INTO`.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

Insira `{'PersonId':'abc123'}` como o primeiro elemento na lista `Owners.SecondaryOwners` existente em um documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

Anexe vários pares de nome-valor à lista `Owners.SecondaryOwners` existente em um documento.

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

```
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook
- Python: Tutorial de início rápido | Referência de Cookbook

INSERTcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `INSERT` comando para adicionar um ou mais documentos Amazon Ion a uma tabela.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno

- Exemplos
- Executar programaticamente usando o driver

Sintaxe

Insira um único documento.

```
INSERT INTO table_name VALUE document
```

Insira vários documentos.

```
INSERT INTO table_name << document, document, ... >>
```

Parâmetros

table_name

O nome da tabela de usuário na qual você deseja qual inserir os dados. A tabela já deve existir. DMLas declarações só são suportadas na visualização padrão do usuário.

document

Um QLDBdocumento válido. Você deve especificar pelo menos um documento. Vários documentos devem ser separados por vírgulas.

O documento deve ser indicado por chaves (`{...}`).

Cada nome do campo no documento é um símbolo de íon que diferencia entre maiúsculas e minúsculas e pode ser indicado por aspas simples no PartiQL (`'...'`).

Os valores de string também são denotados com aspas simples (`'...'`) em PartiQL.

Qualquer literal de Ion pode ser indicado com acentos graves (`` `...` ``).

> **Note**
>
> Os colchetes angulares duplos (`<<...>>`) denotam uma coleção não ordenada (conhecida como bolsa no PartiQL) e são necessários somente se você quiser inserir vários documentos.

Valor de retorno

`documentId`: o ID exclusivo de cada documento que você inseriu.

Exemplos

Insira um único documento.

```
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`
}
```

Essa instrução retorna o ID exclusivo do documento que você inseriu, da seguinte forma.

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

Insira vários documentos.

```
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'
}
>>
```

Essa instrução retorna o ID exclusivo de cada documento que você inseriu, da seguinte forma.

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook

- Python: Tutorial de início rápido | Referência de Cookbook

SELECTcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `SELECT` comando para recuperar dados de uma ou mais tabelas. Cada `SELECT` consulta QLDB é processada em uma transação e está sujeita a um limite de tempo limite de transação.

A ordem dos resultados não é específica e pode variar para cada consulta `SELECT`. Você não deve confiar na ordem dos resultados para qualquer consulta emQLDB.

Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

> **Warning**
>
> Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. O partiQL suporta essas consultas porque é compatível. SQL No entanto, não execute varreduras de tabela para casos de uso de produção emQLDB. Verificações de tabela podem causar problemas de performance em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade em um campo indexado ou em um ID de documento, por exemplo, `WHERE indexedField = 123` ou `WHERE indexedField IN (456, 789)`. Para ter mais informações, consulte Otimizar a performance da consulta.

Tópicos

- Sintaxe
- Parâmetros

- Junções
- Limitações de consulta aninhada
- Exemplos
- Executar programaticamente usando o driver

Sintaxe

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
[ WHERE condition ]
```

Parâmetros

VALUE

Um qualificador para sua expressão que faz com que a consulta retorne o valor do tipo de dados bruto, em vez de o valor ser encapsulado em uma estrutura de tuplo.

expression

Uma projeção formada a partir do curinga `*` ou uma lista de projeção de um ou mais campos de documentos do conjunto de resultados. Uma expressão pode consistir em chamadas para Funções PartiQL ou campos que são modificados por operadores PartiQL.

AS *field_alias*

(Opcional) Um alias temporário definido pelo usuário para o campo usado no conjunto de resultados final. A palavra-chave `AS` é opcional.

Se você não especificar um alias para uma expressão que não for um nome de campo simples, o conjunto de resultados aplicará um nome padrão à coluna.

FROM *source*

Uma origem a ser consultada. As únicas origens atualmente suportadas são nomes de tabelas, junções internas entre tabelas, consultas `SELECT` aninhadas (sujeitas aLimitações de consulta aninhada) e chamadas de função de histórico para uma tabela.

Você deve especificar pelo menos uma origem. Várias origens devem ser separadas por vírgulas.

AS ***source_alias***

(Opcional) Um alias definido pelo usuário que varia em uma origem a ser consultada. Todos os aliases de origem usados na cláusula `SELECT` OU `WHERE` devem ser declarados na cláusula `FROM`. A palavra-chave `AS` é opcional.

AT ***idx_alias***

(Opcional) Um alias definido pelo usuário que se vincula ao número de índice (ordinal) de cada elemento em uma lista da origem. O alias deve ser declarado na cláusula `FROM` usando a palavra-chave `AT`.

BY ***id_alias***

(Opcional) Um alias definido pelo usuário que se vincula ao campo de metadados `id` de cada documento no conjunto de resultados. O alias deve ser declarado na cláusula `FROM` usando a palavra-chave `BY`. Isso é útil quando você deseja projetar ou filtrar o ID do documento ao consultar a visualização padrão do usuário. Para ter mais informações, consulte Usar a cláusula BY para consultar o ID do documento.

WHERE ***condition***

Os critérios de seleção e os critérios de junção (se aplicável) para a consulta.

> **Note**
>
> Se você omitir a cláusula `WHERE`, todos os documentos na tabela serão restaurados.

Junções

No momento, há suporte apenas para junções internas. Você pode escrever consultas de junção interna usando a cláusula `INNER JOIN` explícita, da seguinte maneira. Nessa sintaxe, `JOIN` deve estar emparelhado com `ON`, e a palavra-chave `INNER` é opcional.

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

Ou você pode escrever junções internas usando a sintaxe implícita, da seguinte maneira.

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

Limitações de consulta aninhada

Você pode escrever consultas aninhadas (subconsultas) dentro de `SELECT` expressões e origens `FROM`. A principal restrição é que somente a consulta mais externa pode acessar o ambiente de banco de dados global. Por exemplo, suponha que você tenha um ledger com tabelas `VehicleRegistration` e `Person`. A consulta aninhada a seguir não é válida porque a interna `SELECT` tenta acessar `Person`.

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Já a consulta aninhada a seguir é válida.

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Exemplos

A consulta a seguir mostra um caractere curinga `SELECT` básico com uma cláusula de predicado `WHERE` padrão que usa o operado `IN`.

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

A seguir, são mostradas projeções `SELECT` com um filtro de string de caracteres.

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

O exemplo a seguir mostra uma subconsulta correlacionada que nivela os dados aninhados. Observe que o caractere `@` é tecnicamente opcional aqui. Mas isso indica explicitamente que você

deseja a estrutura `Owners` que está aninhada em `VehicleRegistration`, não uma coleção com um nome `Owners` diferente (se houver). Para obter mais contexto, consulte Dados aninhados no capítulo Trabalhar com dados e histórico.

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

Veja a seguir uma subconsulta na lista `SELECT` que projeta dados aninhados, além de uma junção interna.

```
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')
```

O exemplo a seguir mostra uma junção interna explícita.

```
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')
```

Veja a seguir uma projeção do campo de `id` metadados do documento, usando a cláusula BY.

```
SELECT
    r_id,
```

```
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
    r_id = 'documentId'
```

O exemplo usa a cláusula BY para unir as tabelas `DriversLicense` e `Person` em seus `PersonId` e campos do documento `id`, respectivamente.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

O exemplo usa Visão confirmada para unir as tabelas `DriversLicense` e `Person` em seus `PersonId` e campos do documento `id`, respectivamente.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

O exemplo a seguir retorna o número de índice `PersonId` e (ordinal) de cada pessoa na lista `Owners.SecondaryOwners` para um documento na tabela `VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook
- Python: Tutorial de início rápido | Referência de Cookbook

UPDATEcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `UPDATE` comando para modificar o valor de um ou mais elementos em um documento. Se um elemento não existir, ele será inserido.

Você também pode usar esse comando para inserir e remover explicitamente elementos específicos em um documento, de forma semelhante às instruções FROM(INSERT,REMOVE, ouSET).

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno
- Exemplos
- Executar programaticamente usando o driver

Sintaxe

UPDATE-SET

Atualize um ou mais elementos em um documento. Se um elemento não existir, ele será inserido. Isso é semanticamente o mesmo que a SET declaração FROM-.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
```

```
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

UPDATE-INSERT

Insira um novo elemento em um documento existente. Para inserir um novo documento de nível superior em uma tabela, você deve usar INSERT.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

UPDATE-REMOVE

Remova um elemento existente em um documento ou remova um documento inteiro de nível superior. A última é semanticamente igual à sintaxe DELETE tradicional.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

Parâmetros

table_name

O nome do usuário da tabela que contém os dados a serem modificados. DMLas declarações só são suportadas na visualização padrão do usuário. Cada instrução só pode ser executada em uma única tabela.

AS ***table_alias***

(Opcional) Um alias definido pelo usuário que varia em uma tabela a ser atualizada. A palavra-chave `AS` é opcional.

BY ***id_alias***

(Opcional) Um alias definido pelo usuário que se vincula ao campo de metadados `id` de cada documento no conjunto de resultados. O alias deve ser declarado na cláusula `UPDATE` usando a palavra-chave `BY`. Isso é útil quando você deseja filtrar o ID do documento ao consultar a visualização padrão do usuário. Para ter mais informações, consulte Usar a cláusula BY para consultar o ID do documento.

element

Um elemento de documento a ser criado ou modificado.

data

Um novo valor para o elemento.

AT ***key_name***

Um nome da chave a ser adicionado aos documentos a serem modificados. Você deve especificar o `VALUE` correspondente junto com o nome da chave. Isso é necessário para inserir um novo valor `AT` em uma posição específica em um documento.

WHERE ***condition***

Os critérios de seleção para os documentos a serem modificados.

> **Note**
>
> Se você omitir a cláusula `WHERE`, todos os documentos na tabela serão modificados.

Valor de retorno

`documentId`: o ID exclusivo de cada documento que você atualizou.

Exemplos

Atualize um campo em um documento. Se o campo não existir, ele será inserido.

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

Filtre no campo de metadados `id` do documento atribuído pelo sistema.

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

Substitua um documento inteiro.

```
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'
```

Modifica o campo `PersonId` do primeiro elemento na lista `Owners.SecondaryOwners` em um documento.

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

Insere `{'Mileage':26500}` como um par nome-valor de nível superior em um documento na tabela `Vehicle`.

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

Anexa `{'PersonId':'abc123'}` como um par nome-valor no campo `Owners.SecondaryOwners` de um documento na tabela `VehicleRegistration`. Observe que `Owners.SecondaryOwners` já deve existir e ser um tipo de dados da lista para que essa instrução seja válida. Caso contrário, a palavra-chave `AT` é exigida na cláusula `INSERT INTO`.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

Insira `{'PersonId':'abc123'}` como o primeiro elemento na lista `Owners.SecondaryOwners` existente em um documento.

```
UPDATE VehicleRegistration AS r
```

```
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

Anexe vários pares de nome-valor à lista `Owners.SecondaryOwners` existente em um documento.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

Remove um elemento existente em um documento.

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

Remove um documento inteiro de uma tabela.

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

Remove o primeiro elemento da lista `Owners.SecondaryOwners` em um documento na tabela `VehicleRegistration`.

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

Executar programaticamente usando o driver

Para saber como executar programaticamente essa instrução usando o QLDB driver, consulte os seguintes tutoriais em Introdução ao driver:

- Java: Tutorial de início rápido | Referência de Cookbook
- .NET: Tutorial de início rápido | Referência de Cookbook
- Go: Tutorial de início rápido | Referência de Cookbook
- Node.js: Tutorial de início rápido | Referência de Cookbook

- Python: Tutorial de início rápido | Referência de Cookbook

UNDROPTABLEcomando na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use o `UNDROP TABLE` comando para reativar uma tabela que você descartou anteriormente (ou seja, desativou). Desativar ou reativar uma tabela não afeta seus documentos ou índices.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Sintaxe
- Parâmetros
- Valor de retorno
- Exemplos

Sintaxe

```
UNDROP TABLE "tableId"
```

Parâmetros

"*tableId*"

O ID exclusivo da tabela a ser reativada, indicado por aspas duplas.

A tabela deve ter sido descartada anteriormente, o que significa que ela existe na tabela do catálogo do sistema `information_schema.user_tables` e tem o status de`INACTIVE`. Também não deve haver nenhuma tabela ativa existente com o mesmo nome.

Valor de retorno

`tableId`: o ID exclusivo da tabela que você reativou.

Exemplos

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Funções partiQL na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O PartiQL na Amazon QLDB oferece suporte às seguintes variantes integradas de funções SQL padrão.

> **Note**
>
> No momento, não há suporte para quaisquer SQL funções que não estejam incluídas nessa listaQLDB.
> Essa referência de função é baseada na Funções Integradas na documentação PartiQL.

Propagação de tipo desconhecido (nulo e ausente)

Salvo indicação em contrário, todas essas funções propagam valores de argumentos nulos e ausentes. A propagação de `NULL` ou `MISSING` é definida como retorno `NULL` se algum argumento de função for `NULL` ou`MISSING`. A seguir estão exemplos dessa propagação.

```
CHAR_LENGTH(null)    -- null
```

```
CHAR_LENGTH(missing) -- null (also returns null)
```

Funções agregadas

- AVG
- COUNT
- MAX
- MIN
- SIZE
- SUM

Funções condicionais

- COALESCE
- EXISTS
- NULLIF

Perfis de data e hora

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

Funções escalares

- TXID

Funções de string

- CHAR_LENGTH
- CHARACTER_LENGTH
- LOWER

- SUBSTRING
- TRIM
- UPPER

Funções de formatação de tipo de dados

- CAST
- PARA_ STRING
- PARA_ TIMESTAMP

AVGfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `AVG` função para retornar a média (média aritmética) dos valores da expressão de entrada. Essa função funciona com valores numéricos e ignora valores nulos ou ausentes.

Sintaxe

```
AVG ( expression )
```

Argumentos

`expression`

O nome do campo ou expressão do tipo de dados numéricos no qual a função opera.

Tipos de dados

Tipos de argumento com suporte:

- `int`
- `decimal`
- `float`

Tipo de retorno: `decimal`

Exemplos

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

Funções relacionadas

- COUNT
- MAX
- MIN
- SIZE
- SUM

CASTfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `CAST` função para avaliar uma determinada expressão em um valor e converter o valor em um tipo de dados de destino especificado. Se a conversão não puder ser feita, a função retornará um erro.

Sintaxe

```
CAST ( expression AS type )
```

Argumentos

expression

O nome do campo ou expressão avaliada como um valor que a função converte. A conversão de valores nulos retorna nulos. Esse parâmetro pode ser qualquer um dos Tipos de dados com suporte.

type

O nome do tipo de dados de destino para conversão. Esse parâmetro pode ser um dos Tipos de dados com suporte.

Tipo de retorno

O tipo de dados especificado pelo *type* argumento.

Exemplos

Os exemplos a seguir mostram a propagação de tipos desconhecidos (`NULL` ou `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)
```

Qualquer valor que não seja de um tipo desconhecido não pode ser convertido em `NULL` ou`MISSING`.

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

Os exemplos a seguir mostram o lançamento de `AS boolean`.

```
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'`)
```

Os exemplos a seguir mostram o lançamento de `AS integer`.

```
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
```

Os exemplos a seguir mostram o lançamento de `AS float`.

```
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
```

Os exemplos a seguir mostram o lançamento de `AS decimal`.

```
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
```

Os exemplos a seguir mostram o lançamento de `AS timestamp`.

```
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
```

Os exemplos a seguir mostram o lançamento de `AS symbol`.

```
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'
```

Os exemplos a seguir mostram o lançamento de `AS string`.

```
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"
```

Os exemplos a seguir mostram o lançamento de `AS struct`.

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
```

```
CAST(true       AS struct) -- err
```

Os exemplos a seguir mostram o lançamento de `AS list`.

```
CAST(`[1, 2, 3]`        AS list) -- [1,2,3]
CAST(<<'a', { 'b':2 }>> AS list) -- ["a",{'b':2}]
CAST({ 'b':2 }          AS list) -- error
```

Os exemplos a seguir são instruções executáveis que incluem alguns dos exemplos anteriores.

```
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"
```

Funções relacionadas

- PARA_ STRING
- PARA_ TIMESTAMP

CHAR_ LENGTH função na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `CHAR_LENGTH` função para retornar o número de caracteres na string especificada, em que o caractere é definido como um único ponto de código Unicode.

Sintaxe

```
CHAR_LENGTH ( string )
```

`CHAR_LENGTH` é sinônimo de CHARACTER_ LENGTH função na Amazon QLDB.

Argumentos

`string`

O nome do campo ou a expressão do tipo de dados `string` que a função avalia.

Tipo de retorno

`int`

Exemplos

```
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)
```

Funções relacionadas

- LOWER
- SUBSTRING
- TRIM
- UPPER

CHARACTER_ LENGTH função na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Sinônimo da função `CHAR_LENGTH`.

Consulte CHAR_ LENGTH função na Amazon QLDB.

COALESCEfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, com uma lista de um ou mais argumentos, use a `COALESCE` função para avaliar os argumentos na ordem da esquerda para a direita e retornar o primeiro valor que não seja de um tipo desconhecido (`NULL`ou`MISSING`). Se todos os tipos de argumentos forem desconhecidos, o resultado será `NULL`.

A função `COALESCE` não propaga `NULL` e `MISSING`.

Sintaxe

```
COALESCE ( expression [, expression, ... ] )
```

Argumentos

`expression`

A lista de um ou mais nomes de campo ou expressões que a função avalia. Cada argumento pode ser qualquer um dos Tipos de dados com suporte.

Tipo de retorno

Qualquer tipo de dados com suporte. O tipo de retorno é `NULL` ou igual ao tipo da primeira expressão que é avaliada como um valor não nulo e não ausente.

Exemplos

```
SELECT COALESCE(1, null) FROM << 0 >>         -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>   -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>  -- "string"
```

Funções relacionadas

- EXISTS
- NULLIF

COUNTfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `COUNT` função para retornar o número de documentos definidos pela expressão especificada. A função tem duas variações:

- `COUNT(*)`: conta todos os documentos na tabela de destino, independentemente de incluírem ou não valores nulos ou ausentes.
- `COUNT(expression)`: calcula o número de documentos com valores não nulos em um campo ou expressão específica existente.

> **Warning**
>
> A função `COUNT` não está otimizada, portanto, não recomendamos usá-la sem uma pesquisa indexada. Quando você executa uma consulta QLDB sem uma pesquisa indexada, ela invoca uma verificação completa da tabela. Isso pode causar problemas de desempenho em tabelas grandes, incluindo conflitos de simultaneidade e tempos limite de transação.
> Para evitar verificações de tabelas, você deve executar instruções com uma cláusula de predicado `WHERE` usando um operador de igualdade (= ou `IN`) em um campo indexado ou em um ID de documento. Para ter mais informações, consulte Otimizar a performance da consulta.

Sintaxe

```
COUNT ( * | expression )
```

Argumentos

expression

O nome do campo ou expressão na qual a função opera. Esse parâmetro pode ser qualquer um dos Tipos de dados com suporte.

Tipo de retorno

`int`

Exemplos

```
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
```

Funções relacionadas

- AVG
- MAX
- MIN
- SIZE
- SUM

DATE_ ADD função na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `DATE_ADD` função para incrementar um determinado valor de timestamp em um intervalo especificado.

Sintaxe

```
DATE_ADD( datetimepart, interval, timestamp )
```

Argumentos

datetimepart

A parte da data ou hora na qual a função opera. Esse parâmetro pode ser:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

interval

O número inteiro que especifica o intervalo a ser adicionado ao dado *timestamp*. Um número inteiro negativo subtrai o intervalo.

timestamp

O nome do campo ou a expressão do tipo de dados `timestamp` que a função incrementa.

Um valor literal de timestamp de Ion pode ser indicado com backticks (`` `...` ``). Para obter detalhes do formato e exemplos de valores de carimbo de data/hora, consulte Timestamps no documento de especificação do Amazon Ion.

Tipo de retorno

`timestamp`

Exemplos

```
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
```

Funções relacionadas

- DATE_DIFF
- EXTRACT
- PARA_ STRING
- PARA_ TIMESTAMP
- UTCNOW

DATE_ DIFF função na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `DATE_DIFF` função para retornar a diferença entre as partes de data especificadas de dois carimbos de data/hora fornecidos.

Sintaxe

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

Argumentos

datetimepart

A parte da data ou hora na qual a função opera. Esse parâmetro pode ser:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

timestamp1, *timestamp2*

O nome do campo ou a expressão do tipo de dados `timestamp` que a função compara. If (Se) *timestamp2* é posterior a *timestamp1*, o resultado é positivo. If (Se) *timestamp2* é anterior ao *timestamp1*, o resultado é negativo.

Um valor literal de timestamp de Ion pode ser indicado com backticks (`` `...` ``). Para obter detalhes do formato e exemplos de valores de carimbo de data/hora, consulte Timestamps no documento de especificação do Amazon Ion.

Tipo de retorno

`int`

Exemplos

```
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
```

Funções relacionadas

- DATE_ADD
- EXTRACT
- PARA_ STRING
- PARA_ TIMESTAMP
- UTCNOW

EXISTSfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, com um valor PartiQL, use a `EXISTS` função para retornar `TRUE` se o valor for uma coleção não vazia. Caso contrário, essa função retornará `FALSE`. Se a entrada para `EXISTS` não for um contêiner, o resultado será `FALSE`.

A função `EXISTS` não propaga `NULL` e `MISSING`.

Sintaxe

```
EXISTS ( value )
```

Argumentos

`value`

O nome do campo ou a expressão que a função avalia. Esse parâmetro pode ser qualquer um dos Tipos de dados com suporte.

Tipo de retorno

`bool`

Exemplos

```
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
```

Funções relacionadas

- COALESCE
- NULLIF

EXTRACTfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `EXTRACT` função para retornar o valor inteiro de uma data ou hora especificada de um determinado timestamp.

Sintaxe

```
EXTRACT ( datetimepart FROM timestamp )
```

Argumentos

datetimepart

A parte da data ou hora na qual a função extrai. Esse parâmetro pode ser:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`
- `timezone_hour`
- `timezone_minute`

timestamp

O nome do campo ou a expressão do tipo de dados `timestamp` que a função extrai. Se esse parâmetro for de um tipo desconhecido (`NULL`ou`MISSING`), a função retornará `NULL`.

Um valor literal de timestamp de Ion pode ser indicado com backticks (`` `...` ``). Para obter detalhes do formato e exemplos de valores de carimbo de data/hora, consulte Timestamps no documento de especificação do Amazon Ion.

Tipo de retorno

`int`

Exemplos

```
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
```

Funções relacionadas

- DATE_ADD
- DATE_DIFF
- PARA_ STRING
- PARA_ TIMESTAMP
- UTCNOW

LOWERfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `LOWER` função para converter todos os caracteres maiúsculos em minúsculos em uma determinada string.

Sintaxe

```
LOWER ( string )
```

Argumentos

`string`

O nome do campo ou a expressão do tipo de dados `string` que a função converte.

Tipo de retorno

`string`

Exemplos

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

Funções relacionadas

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

MAXfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `MAX` função para retornar o valor máximo em um conjunto de valores numéricos.

Sintaxe

```
MAX ( expression )
```

Argumentos

`expression`

O nome do campo ou expressão do tipo de dados numéricos no qual a função opera.

Tipos de dados

Tipos de argumento com suporte:

- `int`
- `decimal`
- `float`

Tipos de devoluções compatíveis:

- `int`
- `decimal`
- `float`

Exemplos

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

Funções relacionadas

- AVG
- COUNT
- MIN
- SIZE
- SUM

MINfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `MIN` função para retornar o valor mínimo em um conjunto de valores numéricos.

Sintaxe

```
MIN ( expression )
```

Argumentos

expression

O nome do campo ou expressão do tipo de dados numéricos no qual a função opera.

Tipos de dados

Tipos de argumento com suporte:

- `int`
- `decimal`
- `float`

Tipos de devoluções compatíveis:

- `int`
- `decimal`
- `float`

Exemplos

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

Funções relacionadas

- AVG
- COUNT
- MAX

- SIZE
- SUM

NULLIFfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, dadas duas expressões, use a `NULLIF` função para retornar `NULL` se as duas expressões tiverem o mesmo valor. Caso contrário, essa função retornará o resultado da avaliação da primeira expressão.

A função `NULLIF` não propaga `NULL` e `MISSING`.

Sintaxe

```
NULLIF ( expression1, expression2 )
```

Argumentos

`expression1`, *`expression2`*

O dois nomes do campo ou a expressão do tipo de dados que a função compara. Esses parâmetros podem ser qualquer um dos Tipos de dados com suporte.

Tipo de retorno

Qualquer tipo de dados com suporte. O tipo de retorno é `NULL` ou igual ao tipo da primeira expressão.

Exemplos

```
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
```

Funções relacionadas

- COALESCE
- EXISTS

SIZEfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `SIZE` função para retornar o número de elementos em um determinado tipo de dados de contêiner (lista, estrutura ou bolsa).

Sintaxe

```
SIZE ( container )
```

Argumentos

`container`

O nome do campo ou expressão do contêiner no qual a função opera.

Tipos de dados

Tipos de argumento com suporte:

- list
- estrutura
- bolsa

Tipo de retorno: `int`

Se a entrada para `SIZE` não for um contêiner, a função gerará um erro.

Exemplos

```
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
```

Funções relacionadas

- AVG
- COUNT
- MAX
- MIN
- SUM

SUBSTRINGfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `SUBSTRING` função para retornar uma substring de uma determinada string. A substring começa no índice de início especificado e termina no último caractere da string ou no comprimento especificado.

Sintaxe

```
SUBSTRING ( string, start-index [, length ] )
```

Argumentos

`string`

O nome do campo ou a expressão do tipo de dados `string` do qual extrair uma substring.

`start-index`

A posição inicial dentro do *`string`* a partir do qual iniciar a extração. Esse número pode ser negativo.

O primeiro personagem de *`string`* tem um índice de 1.

`length`

(Opcional) O número de caracteres (pontos de código) a serem extraídos do *`string`*, começando em *`start-index`* e terminando em (*`start-index`* + *`length`*) - 1. Em outras palavras, o comprimento da substring. Esse número não pode ser negativo.

Se esse parâmetro não for fornecido, a função prosseguirá até o final do *`string`*.

Tipo de retorno

`string`

Exemplos

```
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"
```

Funções relacionadas

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

SUMfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `SUM` função para retornar a soma dos valores do campo de entrada ou da expressão. Essa função funciona com valores numéricos e ignora valores nulos ou ausentes.

Sintaxe

```
SUM ( expression )
```

Argumentos

expression

O nome do campo ou expressão do tipo de dados numéricos no qual a função opera.

Tipos de dados

Tipos de argumento com suporte:

- `int`
- `decimal`
- `float`

Tipos de devoluções compatíveis:

- `int`: para argumentos inteiros
- `decimal`: para argumentos decimais ou de ponto flutuante

Exemplos

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

Funções relacionadas

- AVG
- COUNT
- MAX
- MIN
- SIZE

STRINGFunção TO_ na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `TO_STRING` função para retornar uma representação de string de um determinado timestamp no padrão de formato especificado.

Sintaxe

```
TO_STRING ( timestamp, 'format' )
```

Argumentos

`timestamp`

O nome do campo ou a expressão do tipo de dados `timestamp` que a função converte em uma string.

Um valor literal de timestamp de Ion pode ser indicado com backticks (`` `...` ``). Para obter detalhes do formato e exemplos de valores de carimbo de data/hora, consulte Timestamps no documento de especificação do Amazon Ion.

`format`

A string literal que especifica o padrão de formato do resultado, em termos de suas partes de data. Para obter os formatos válidos, consulte Strings de formato da data e hora.

Tipo de retorno

`string`

Exemplos

```
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"
```

Funções relacionadas

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- PARA_ TIMESTAMP
- UTCNOW

TIMESTAMPFunção TO_ na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, dada uma string que representa um timestamp, use a `TO_TIMESTAMP` função para converter a string em um tipo de `timestamp` dados. Esta é a operação inversa de `TO_STRING`.

Sintaxe

```
TO_TIMESTAMP ( string [, 'format' ] )
```

Argumentos

string

O nome do campo ou a expressão do tipo de dados `string` que a função converte em um timestamp.

format

(Opcional) A string literal que define o padrão de formato da entrada *string*, em termos de suas partes de data. Para obter os formatos válidos, consulte Strings de formato da data e hora.

Se esse argumento for omitido, a função assume que o *string* está no formato de um carimbo de data/hora padrão do Ion. Essa é a maneira recomendada de analisar um timestamp do Ion usando essa função.

O preenchimento zero é opcional ao usar um símbolo de formato de caractere único (como `y`, `M`, `d`, `H`, `h`, `m`, `s`), mas é obrigatório para suas variantes com preenchimento zero (como `yyyy`, `MM`, `dd`, `HH`, `hh`, `mm`, `ss`).

Tratamento especial é dado aos anos de dois dígitos (símbolo de formato `yy`). 1900 é adicionado a valores maiores que ou iguais a 70 e 2000 é adicionado a valores menores que 70.

Os nomes dos meses e os indicadores AM ou PM não diferenciam maiúsculas de minúsculas.

Tipo de retorno

`timestamp`

Exemplos

```
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
```

Funções relacionadas

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- PARA_ STRING
- UTCNOW

TRIMfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `TRIM` função para cortar uma determinada string removendo os espaços em branco à esquerda e à direita ou um conjunto específico de caracteres.

Sintaxe

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

Argumentos

LEADING

(Opcional) Indica que os espaços em branco ou os caracteres especificados devem ser removidos do início de *`string`*. Se não for especificado, o comportamento padrão será`BOTH`.

TRAILING

(Opcional) Indica que os espaços em branco ou os caracteres especificados devem ser removidos do final de *`string`*. Se não for especificado, o comportamento padrão será`BOTH`.

BOTH

(Opcional) Indica que os espaços em branco à esquerda e à direita ou os caracteres especificados devem ser removidos do início e do final de *`string`*.

`characters`

(Opcional) O conjunto de caracteres a serem removidos, especificado como um `string`.

Se esse parâmetro não for fornecido, os espaços em branco serão removidos.

`string`

O nome do campo ou a expressão do tipo de dados `string` que a função corta.

Tipo de retorno

`string`

Exemplos

```
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         "
```

Funções relacionadas

- CHAR_LENGTH
- LOWER

- SUBSTRING
- UPPER

TXIDfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `TXID` função para retornar o ID exclusivo da transação do extrato atual que você está executando. Esse é o valor atribuído ao campo de metadados `txId` de um documento quando a transação atual é confirmada no diário.

Sintaxe

```
TXID()
```

Argumentos

Nenhum

Tipo de retorno

`string`

Exemplos

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

UPPERfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `UPPER` função para converter todos os caracteres minúsculos em maiúsculas em uma determinada string.

Sintaxe

```
UPPER ( string )
```

Argumentos

`string`

O nome do campo ou a expressão do tipo de dados `string` que a função converte.

Tipo de retorno

`string`

Exemplos

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

Funções relacionadas

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

UTCNOWfunção na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, use a `UTCNOW` função para retornar a hora atual em Tempo Universal Coordenado (UTC) como `timestamp` a.

Sintaxe

```
UTCNOW()
```

Essa função exige que você especifique uma cláusula `FROM` em uma consulta `SELECT`.

Argumentos

Nenhum

Tipo de retorno

`timestamp`

Exemplos

```
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'
```

Funções relacionadas

- DATE_ADD
- DATE_DIFF
- EXTRACT
- PARA_ STRING

- PARA_ TIMESTAMP

Strings de formato da data e hora

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção fornece informações de referência para strings de caracteres de formato da data e hora.

As strings de formato da data e hora se aplicam às funções `TO_STRING` e `TO_TIMESTAMP`. Essas strings de caracteres podem conter separadores de partes de data (como '`-`', '`/`' ou '`:`') e os seguintes símbolos de formato.

Formato	Exemplo	Descrição
`yy`	70	Ano de dois dígitos
`y`	1970	Ano de quatro dígitos.
`yyyy`	1970	Ano com quatro dígitos preenchido com zeros
`M`	1	inteiro do mês
`MM`	01	Inteiro de mês preenchido com zeros
`MMM`	Jan	Nome do mês abreviado
`MMMM`	janeiro	Nome do mês completo
`d`	2	Dia do mês (1 a 31)
`dd`	02	Dia do mês preenchido com zeros (01 a 31)

Formato	Exemplo	Descrição
`a`	AM ou PM	Indicador de meridiano (para relógio de 12 horas)
`h`	3	Hora (relógio de 12 horas, 01–12)
`hh`	03	Hora preenchida com zero (relógio de 12 horas, 01—12)
`H`	3	Hora (relógio de 24 horas, 00–23)
`HH`	03	Hora preenchida com zero (relógio de 24 horas, 01—23)
`m`	4	Minutos (00–59)
`mm`	04	Minutos preenchidos com zero (00—59)
`s`	5	Segundos (00–59)
`ss`	05	Segundos preenchidos com zero (00—59)
`S`	0	Fração de um segundo (precisão: 0,1, intervalo: de 0,0 a 0,9)
`SS`	06	Fração de um segundo (precisão: 0,01, intervalo: de 0,0 a 0,99)
`SSS`	060	Fração de um segundo (precisão: 0,001, intervalo: de 0,0 a 0,999)

Formato	Exemplo	Descrição
X	+07 ou Z	Deslocamento UTC em horas ou “Z” se o deslocamento for 0
XX	+0700 ou Z	Deslocamento UTC em horas e minutos, ou “Z” se o deslocamento for 0
XXX	+ 07:00 ou Z	Deslocamento UTC em horas e minutos, ou “Z” se o deslocamento for 0
x	+07	Compensação a partir de UTC em horas
xx	+0700	Compensação UTC em horas e minutos
xxx	+07:00	Compensação UTC em horas e minutos

Procedimentos armazenados em partiQL na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Na AmazonQLDB, você pode usar o `EXEC` comando para executar procedimentos armazenados do PartiQL na seguinte sintaxe.

```
EXEC stored_procedure_name argument [, ... ]
```

QLDBsuporta somente os seguintes procedimentos armazenados no sistema:

Tópicos

- REDACT_ procedimento REVISION armazenado na Amazon QLDB

REDACT_ procedimento REVISION armazenado na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

> **Note**
>
> No momento, todos os ledgers criados antes de 22 de julho de 2021 não são elegíveis para edição. Você pode ver a hora de criação do seu livro contábil no QLDB console da Amazon.

Na AmazonQLDB, use o procedimento `REDACT_REVISION` armazenado para excluir permanentemente uma revisão de documento individual e inativa no armazenamento indexado e no armazenamento de diário. Esse procedimento armazenado exclui todos os dados do usuário na revisão especificada. No entanto, deixa a sequência do diário e os metadados do documento, incluindo o ID e o hash do documento, inalterados. Essa operação é irreversível.

A revisão do documento especificada deve ser uma revisão inativa no histórico. A revisão ativa mais recente de um documento não está qualificada para redação.

Depois de enviar uma solicitação de redação executando esse procedimento armazenado, QLDB processa a redação dos dados de forma assíncrona. Depois que a redação for concluída, os dados do usuário na revisão especificada (representados pela estrutura `data`) são substituídos por um novo campo `dataHash`. O valor desse campo é o hash de Amazon Ion da estrutura `data` removida. Como resultado, o livro mantém a integridade geral dos dados e permanece criptograficamente verificável por meio das operações de verificação existentes. API

Para ver um exemplo de uma operação de edição com dados de exemplo, consulte Exemplo de redação em Redigir revisões de documentos.

> **Note**
>
> Para aprender a controlar o acesso para executar este comando do PartiQL em tabelas específicas, consulte Introdução ao modo de permissões padrão na Amazon QLDB.

Tópicos

- Considerações e limitações de edição
- Sintaxe
- Argumentos
- Valor de retorno
- Exemplos

Considerações e limitações de edição

Antes de começar com a redação de dados na AmazonQLDB, certifique-se de analisar as seguintes considerações e limitações:

- O procedimento `REDACT_REVISION` armazenado tem como alvo os dados do usuário em uma revisão de documento individual e inativa. Para editar várias revisões, você deve executar o procedimento armazenado uma vez para cada revisão. Você pode editar uma revisão por transação.
- Para redigir campos específicos em uma revisão de documento, você deve usar uma instrução separada de linguagem de manipulação de dados (DML) para modificar a revisão primeiro. Para ter mais informações, consulte Editar um campo específico em uma revisão.
- Depois de QLDB receber uma solicitação de redação, você não poderá cancelar nem alterar a solicitação. Para confirmar se uma edição foi concluída, você pode verificar se a estrutura `data` de uma revisão foi substituída por um campo `dataHash`. Para saber mais, consulte Verificar se uma redação está completa.
- A redação não tem impacto em nenhum QLDB dado que seja replicado fora do QLDB serviço. Isso inclui todas as exportações para o Amazon S3 e fluxos para o Amazon Kinesis Data Streams. Você deve usar outros métodos de retenção de dados para gerenciar quaisquer dados armazenados fora doQLDB.
- A edição não tem impacto nos valores literais em instruções partiQL que são registradas no diário. Como prática recomendada, você deve executar instruções parametrizadas de forma programática

usando espaços reservados de variáveis em vez de valores literais. Um espaço reservado é escrito no diário como um ponto de interrogação (?) em vez de qualquer informação confidencial que possa exigir edição.

Para aprender a executar programaticamente instruções partiQL usando o QLDB driver, consulte os tutoriais de cada linguagem de programação compatível em. Conceitos básicos do driver

Sintaxe

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argumentos

`block-address`

A localização do bloco de diário da revisão do documento a ser editada. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Esse é um valor literal de Ion indicado por acentos graves. Por exemplo: .

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

Para saber como encontrar o endereço do bloco, consulte Consultar metadados do documento.

'*table-id*'

O ID exclusivo da tabela cuja revisão do documento você deseja redigir, indicada por aspas simples.

Para saber como encontrar o ID da tabela, consulte Consultar o catálogo do sistema.

'*document-id*'

O ID exclusivo do documento cuja revisão será redigida, indicada por aspas simples.

Para saber como encontrar o ID do documento, consulte Consultar metadados do documento.

Valor de retorno

Uma estrutura Amazon Ion que representa a revisão do documento a ser editada, no seguinte formato.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

Campos da estrutura de retorno

- `blockAddress`: a localização do bloco de diário da revisão a ser editada. Um endereço tem os dois campos a seguir:
  - `strandId`: o ID exclusivo da cadeia do diário que contém o bloco.
  - `sequenceNo`: o número do índice que especifica a localização do bloco dentro da cadeia.
- `tableId`: o ID exclusivo da tabela cuja revisão você está editando.
- `documentId`: o ID exclusivo do documento da revisão a ser editada.
- `version`: o número da versão do documento da revisão a ser editada.

Veja a seguir um exemplo da estrutura de retorno com dados de exemplo.

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

Exemplos

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

Operadores partiQL na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

O PartiQL na Amazon QLDB oferece suporte aos seguintes operadores SQLpadrão.

> **Note**
>
> No momento, não há suporte para SQL operadores que não estejam incluídos nessa listaQLDB.

Operadores aritméticos

Operador	Descrição
+	Adicionar
-	Subtrair
*	Multiplicar
/	Dividir
%	Módulo

Operadores de comparação

Operador	Descrição
=	Igual a

Operador	Descrição
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
<>	Não é igual a

Operadores lógicos

Operador	Descrição
`AND`	`TRUE` se todas as condições separadas por `AND` forem `TRUE`
`BETWEEN`	`TRUE` se o operando estiver dentro do intervalo de comparações
`IN`	`TRUE` se o operando for igual a uma expressão em uma lista de expressões
`IS`	`TRUE` se o operando for um determinado tipo de dados PartiQL, incluindo `NULL` ou `MISSING`
`LIKE`	`TRUE` se o operando corresponder a um padrão
`NOT`	Reverte o valor de uma determinada expressão booleana
`OR`	`TRUE` se qualquer uma das condições separadas por `OR` for `TRUE`

Operadores de string

Operador	Descrição
`||`	Concatena duas strings em qualquer dos lados do operador `||` e retorna a string concatenada. Se uma ou ambas as expressões for `NULL`, o resultado da concatenação será nulo.

Palavras-chave reservadas na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A seguir está uma lista de palavras-chave reservadas do PartiQL na Amazon. QLDB Você pode usar uma palavra-chave reservada como identificador entre aspas duplas (por exemplo,`"user"`). Para obter informações sobre as convenções de citação do PartiQL em, consulte. QLDB Consultar o Ion com o PartiQL

> **Important**
>
> As palavras-chave nessa lista são todas consideradas reservadas porque o partiQL é compatível com versões anteriores de -92. SQL No entanto, suporta QLDB apenas um subconjunto dessas palavras reservadas. Para ver a lista de SQL palavras-chave que QLDB atualmente oferecem suporte, consulte os tópicos a seguir:
>
> - Funções PartiQL
> - operadores PartiQL
> - Comandos PartiQL

```
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
```

Referência de formato de dados Amazon Ion na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A Amazon QLDB usa um modelo de notação de dados que unifica o Amazon Ion com um subconjunto de tipos partiQL. Esta seção fornece uma visão geral de referência do formato de dados do documento Ion, separada de sua integração com o partiQL.

Consultando o Ion com o PartiQL na Amazon QLDB

Para saber a sintaxe e a semântica da consulta de dados do Ion com o PartiQL in, QLDB consulte a referência do Consultar o Ion com o PartiQL Amazon PartiQL. QLDB

Para exemplos de código que consultam e processam dados de íons em um QLDB livro contábil, consulte Exemplos de código do Amazon Ion e. Como trabalhar com o Amazon Ion

Tópicos

- O que é o Amazon Ion?
- Especificação de Ion
- JSONcompatível
- Extensões de JSON
- Exemplo de texto Ion
- APIreferências
- Exemplos de códigos Amazon Ion em QLDB

O que é o Amazon Ion?

O Ion é um formato de serialização de dados hierárquico, de código aberto, ricamente tipado, autodescritivo e originalmente desenvolvido internamente na Amazon. É baseado em um modelo de dados abstrato que permite armazenar dados estruturados e não estruturados. É um superconjunto deJSON, o que significa que qualquer JSON documento válido também é um documento Ion válido. Este guia pressupõe um conhecimento prático básico de. JSON Se você ainda não estiver familiarizadoJSON, consulte Introdução JSON para obter mais informações.

Você pode anotar documentos Ion de forma intercambiável em formato de texto legível por humanos ou em formato codificado binário. Por exemploJSON, o formulário de texto é fácil de ler e escrever, suportando prototipagem rápida. A codificação binária é mais compacta e eficiente para persistir, transmitir e analisar. Um processador Ion pode transcodificar entre os dois formatos para representar exatamente o mesmo conjunto de estruturas de dados sem perda de dados. Esse atributo permite que os aplicativos otimizem a forma como processam dados para diferentes casos de uso.

> **Note**
>
> O modelo de dados Ion é estritamente baseado em valores e não oferece suporte a referências. Assim, o modelo de dados pode representar hierarquias de dados que podem ser aninhadas em profundidade arbitrária, mas não em gráficos direcionados.

Especificação de Ion

Para obter uma lista completa dos tipos de dados principais do Ion com descrições completas e detalhes de formatação de valores, consulte o documento de especificação do Ion no GitHub site da Amazon.

Para simplificar o desenvolvimento de aplicativos, o Amazon Ion fornece bibliotecas de clientes que processam dados do Ion para você. Para exemplos de código de casos de uso comuns para processamento de dados de íons, consulte o Amazon Ion Cookbook em GitHub.

JSONcompatível

Da mesma formaJSON, você compõe documentos do Amazon Ion com um conjunto de tipos de dados primitivos e um conjunto de tipos de contêineres definidos recursivamente. O Ion inclui os seguintes tipos JSON de dados tradicionais:

- `null`: um valor genérico nulo (vazio) não digitado. Além disso, conforme descrito na seção a seguir, o Ion suporta um tipo nulo distinto para cada tipo primitivo.
- `bool`: Valores boleanos
- `string`: Literais de texto em Unicode
- `list`: Coleções heterogêneas ordenadas de valores
- `struct`: Coleções não ordenadas de pares nome-valor Por exemploJSON, `struct` permite vários valores por nome, mas isso geralmente é desencorajado.

Extensões de JSON

Tipos de números

Em vez do JSON `number` tipo ambíguo, o Amazon Ion define estritamente os números como um dos seguintes tipos:

- `int`: Números inteiros assinados de tamanho arbitrário
- `decimal`: Números reais codificados em decimal de precisão arbitrária
- `float`: números de ponto flutuante codificados em binário (64 bits). IEEE

Ao analisar documentos, um processador Ion atribui tipos de números da seguinte forma:

- `int`: Números sem expoente ou ponto decimal (por exemplo, `100200`).
- `decimal`: Números sem expoente ou ponto decimal (por exemplo, `0.00001`, `200.0`).
- `float`: Números com um expoente, como notação científica ou notação eletrônica (por exemplo, `2e0`, `3.1e-4`).

Novos tipos de dados

O Amazon Ion adiciona os seguintes tipos de dados:

- `timestamp`: Momentos de data/hora/fuso horário de precisão arbitrária
- `symbol`: Átomos simbólicos Unicode (identificadores)
- `blob`: Dados binários de codificação definida pelo usuário.
- `clob`: Dados de texto de codificação definida pelo usuário
- `sexp`: coleções ordenadas de valores com semântica definida pelo aplicativo.

Tipos nulos

Além do tipo nulo genérico definido porJSON, o Amazon Ion oferece suporte a um tipo nulo distinto para cada tipo primitivo. Isso indica falta de valor e, ao mesmo tempo, mantém um tipo de dados rigoroso.

```
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
```

Exemplo de texto Ion

```
// 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
    ],
}
```

APIreferências

- ion-go
- ion-java
- ion-js
- ion-python

Exemplos de códigos Amazon Ion em QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção fornece exemplos de código que processam dados do Amazon Ion lendo e gravando valores de documentos em um QLDB livro contábil da Amazon. Os exemplos de código usam o QLDB driver para executar instruções partiQL no livro contábil. Esses exemplos fazem parte do aplicativo de amostra em Começando a usar a Amazon QLDB usando um exemplo de tutorial de aplicativo e são de código aberto no GitHub site AWS Samples.

Para exemplos gerais de código que mostram casos de uso comuns de processamento de dados de íons, consulte o Amazon Ion Cookbook on GitHub.

Executar o código

O código do tutorial para cada linguagem de programação segue as seguintes etapas:

1. Conecte-se ao ledger `vehicle-registration` de amostra.
2. Crie uma tabela chamada `IonTypes`.
3. Insira um documento na tabela com um único campo `Name`.
4. Para cada tipo de dados de Ion compatível:
   a. Atualize o campo `Name` do documento com um valor literal do tipo de dados.
   b. Consulte a tabela para obter a revisão mais recente do documento.
   c. Valide se o valor de `Name` manteve suas propriedades de tipo de dados originais verificando se ele corresponde ao tipo esperado.
5. Descarte a tabela `IonTypes`.

> **Note**
>
> Antes de executar esse código de tutorial, você deve criar um ledger chamado `vehicle-registration`.

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()
```

QLDBAPIReferência da Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Este capítulo descreve as API operações de baixo nível da Amazon QLDB que podem ser acessadas por meio HTTP do AWS Command Line Interface (AWS CLI) ou de AWS SDK:

- Amazon QLDB — O gerenciamento de QLDB recursos API (também conhecido como plano de controle). Isso API é usado somente para gerenciar recursos contábeis e para operações de dados não transacionais. Você pode usar essas operações para criar, excluir, descrever, listar e atualizar ledgers. Você também pode verificar os dados do diário criptograficamente e exportar ou transmitir blocos de diário.
- Amazon QLDB Session — Os QLDB dados transacionais. API Você pode usar isso API para executar transações de dados em um livro contábil com instruções partiQL.

> **Important**
>
> Em vez de interagir diretamente com a QLDBsessãoAPI, recomendamos usar o QLDB driver ou o QLDB shell para executar transações de dados em um livro contábil.
>
> - Se você estiver trabalhando com um AWS SDK, use o QLDB driver. O driver fornece uma camada de abstração de alto nível acima dos dados da QLDBsessão API e gerencia a `SendCommand` operação para você. Para obter informações e uma lista das linguagens de programação suportadas, consulte Conceitos básicos do driver.
> - Se você estiver trabalhando com o AWS CLI, use o QLDB shell. O shell é uma interface de linha de comando que usa o QLDB driver para interagir com um livro contábil. Para obter mais informações, consulte Usando o Amazon QLDB shell (APIsomente dados).

Tópicos

- Ações
- Tipos de dados

- Erros comuns
- Parâmetros gerais

Ações

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

As seguintes ações são apoiadas pela AmazonQLDB:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger

- UpdateLedgerPermissionsMode

As seguintes ações são apoiadas pela Amazon QLDB Session:

- SendCommand

Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

As seguintes ações são apoiadas pela AmazonQLDB:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis

- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

CancelJournalKinesisStream

Serviço: Amazon QLDB

Encerra um determinado stream de diários do Amazon QLDB. Antes que um fluxo possa ser cancelado, seu status atual deve ser `ACTIVE`.

Não é possível reiniciar um fluxo depois de tê-lo cancelado. Os recursos de stream do QLDB cancelados estão sujeitos a um período de retenção de 7 dias, portanto, são excluídos automaticamente após a expiração desse limite.

Sintaxe da Solicitação

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

streamId

O UUID (representado em texto codificado em Base62) do stream de diário do QLDB a ser cancelado.

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

StreamId

O UUID (representado em texto codificado em Base62) do stream de diário do QLDB cancelado.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

CreateLedger

Serviço: Amazon QLDB

Cria um novo livro contábil no seu Conta da AWS na região atual.

Sintaxe da Solicitação

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Parâmetros da solicitação de URI

A solicitação não usa nenhum parâmetro de URI.

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

DeletionProtection

Especifica se o ledger está protegido contra exclusão por qualquer usuário. Se ele não for fornecido na criação do razão, esse recurso estará habilitado (`true`) por padrão.

Se a proteção contra exclusão estiver habilitada, você deverá desabilitá-la antes de excluir o razão. Você pode desabilitá-lo chamando a operação `UpdateLedger` para definir esse parâmetro como `false`.

Tipo: booliano

Obrigatório: não

KmsKey

A chave in AWS Key Management Service (AWS KMS) a ser usada para criptografia de dados em repouso no livro contábil. Para obter mais informações, consulte Criptografia em repouso, no Guia do desenvolvedor do Amazon QLDB.

Utilize uma das seguintes opções para especificar esse parâmetro:

- `AWS_OWNED_KMS_KEY`: use uma AWS KMS chave que pertença e seja AWS gerenciada por você.
- Indefinido: por padrão, use uma chave AWS KMS própria.
- Uma chave simétrica do KMS válida e gerenciada pelo cliente: use a chave do KMS de criptografia simétrica, que você cria, detém e gerencia, especificada na sua conta.

  O Amazon QLDB não oferece suporte a chaves assimétricas. Para obter mais informações, consulte Como usar chaves simétricas e assimétricas no Guia do AWS Key Management Service desenvolvedor.

Para especificar uma chave do KMS gerenciada pelo cliente, use o ID da chave, o nome do recurso da Amazon (ARN), o nome do alias ou o ARN do alias. Ao usar um nome de alias, use `"alias/"` como prefixo. Para especificar uma chave em outra Conta da AWS, você deve usar o ARN da chave ou o alias ARN.

Por exemplo: .

- ID da chave: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN da chave: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nome do alias: `alias/ExampleAlias`
- ARN do alias: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Para obter mais informações, consulte Identificadores de chave (KeyId) no Guia do AWS Key Management Service desenvolvedor.

Tipo: sequência

Restrições de tamanho: tamanho máximo de 1600.

Obrigatório: não

Name

O nome do livro contábil que você deseja criar. O nome deve ser exclusivo entre todos os livros contábeis da sua Conta da AWS região atual.

Restrições de nomenclatura para nomes de livros contábeis são definidas em Cotas no Amazon QLDB no Guia do desenvolvedor do Amazon QLDB.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

PermissionsMode

O modo de permissões a ser atribuído ao livro contábil que você deseja criar. Esse parâmetro pode ter um dos valores a seguir:

- `ALLOW_ALL`: um modo de permissões legado que permite o controle do acesso com granularidade em nível de API para razões.

  Esse modo permite aos usuários que tenham a permissão de API `SendCommand` para esse ledger executar todos os comandos PartiQL (portanto, `ALLOW_ALL`) em qualquer tabela no razão especificado. Esse modo desconsidera qualquer política de permissões do IAM em nível de tabela ou comando criada para o razão.
- `STANDARD`: (Recomendado) Um modo de permissões que permite o controle do acesso com granularidade mais fina para razões, tabelas e comandos PartiQL.

  Por padrão, esse modo nega todas as solicitações do usuário para executar comandos do PartiQL em qualquer tabela nesse razão. Para permitir a execução de comandos PartiQL, é necessário criar políticas de permissões do IAM para recursos de tabela específicos e ações PartiQL, além da permissão da API `SendCommand` para o razão. Para saber mais, consulte Conceitos básicos sobre o modo de permissões padrão, no Guia do desenvolvedor do Amazon QLDB.

> **Note**
>
> Recomendamos o uso do modo de permissões `STANDARD` para maximizar a segurança dos dados do seu razão.

Tipo: sequências

Valores Válidos: `ALLOW_ALL | STANDARD`

Obrigatório: Sim

Tags

Os pares de chave-valor a serem adicionados como tags ao ledger que você deseja criar. Chaves de tag fazem distinção entre maiúsculas e minúsculas. Os valores de tag diferenciam maiúsculas de minúsculas e podem ser nulos.

Tipo: mapa de string para string

Entradas do mapa: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de Tamanho de Chave: Tamanho mínimo de 1. O tamanho máximo é 128.

Restrições de tamanho do valor: tamanho mínimo de 0. Tamanho máximo de 256.

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Arn

O nome do recurso da Amazon (ARN) para o ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

CreationDateTime

A data e a hora, no formato epoch, quando o ledger foi criado. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Tipo: carimbo de data/hora

DeletionProtection

Especifica se o ledger está protegido contra exclusão por qualquer usuário. Se ele não for fornecido na criação do razão, esse recurso estará habilitado (`true`) por padrão.

Se a proteção contra exclusão estiver habilitada, você deverá desabilitá-la antes de excluir o razão. Você pode desabilitá-lo chamando a operação `UpdateLedger` para definir esse parâmetro como `false`.

Tipo: booliano

KmsKeyArn

O ARN da chave KMS gerenciada pelo cliente que o ledger usa para criptografia em repouso. Se esse parâmetro for indefinido, o livro contábil usará uma chave AWS KMS própria para criptografia.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Name

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

O modo de permissões a ser atribuído ao ledger que você deseja criar.

Tipo: sequências

Valores Válidos: `ALLOW_ALL | STANDARD`

State

O status atual do ledger.

Tipo: sequências

Valores Válidos: `CREATING | ACTIVE | DELETING | DELETED`

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

LimitExceededException

Você atingiu a quantidade máxima permitida de recursos.

Código de Status HTTP: 400

ResourceAlreadyExistsException

O recurso especificado já existe.

Código de Status HTTP: 409

ResourceInUseException

O recurso especificado não pode ser modificado no momento.

Código de Status HTTP: 409

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteLedger

Serviço: Amazon QLDB

Exclui um ledger e todo o seu conteúdo. Essa ação é irreversível.

Se a proteção contra exclusão estiver ativada, você deverá desabilitá-la antes de excluir o ledger. Você pode desabilitá-lo chamando a operação `UpdateLedger` para definir esse parâmetro como `false`.

Sintaxe da Solicitação

```
DELETE /ledgers/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger que você deseja excluir.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
HTTP/1.1 200
```

Elementos de Resposta

Se a ação tiver êxito, o serviço enviará de volta uma resposta HTTP 200 com um corpo HTTP vazio.

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceInUseException

O recurso especificado não pode ser modificado no momento.

Código de Status HTTP: 409

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DescribeJournalKinesisStream

Serviço: Amazon QLDB

Retorna informações detalhadas sobre o fluxo do diário do Amazon QLDB. O resultado inclui o nome do recurso da Amazon (ARN), nome do stream, hora de criação, status atual e os parâmetros da solicitação original de criação do stream.

Essa ação não retorna nenhum stream de diário expirado. Para obter mais informações, consulte Expiração para fluxos terminais no Guia do desenvolvedor do Amazon QLDB.

Sintaxe da Solicitação

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

streamId

O UUID (representado em texto codificado em Base62) do stream de diário do QLDB a descrever.

Restrições de comprimento: comprimento fixo de 21.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
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"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Stream

Informações sobre o fluxo do diário retornadas por uma solicitação `DescribeJournalS3Export`.

Tipo: objeto JournalKinesisStreamDescription

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DescribeJournalS3Export

Serviço: Amazon QLDB

Retorna informações sobre um trabalho de exportação de diário, incluindo o nome do ledger, a ID de exportação, a hora da criação, o status atual e os parâmetros da solicitação original de criação da exportação.

Essa ação não retorna nenhum trabalho de exportação expirado. Para obter mais informações, consulte Expiração de trabalho de exportação no Guia do desenvolvedor do Amazon QLDB.

Se o trabalho de exportação com o dado `ExportId` não existir, então lança `ResourceNotFoundException`.

Se o livro com o dado `Name` não existir, então lança `ResourceNotFoundException`.

Sintaxe da Solicitação

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

exportId

O UUID (representado em texto codificado em Base62) do trabalho de exportação de diário a descrever.

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
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"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

ExportDescription

Informações sobre o trabalho de exportação do diário retornadas por uma solicitação `DescribeJournalS3Export`.

Tipo: objeto JournalS3ExportDescription

Erros

Para obter informações sobre os erros comuns a todas as ações, consulte Erros comuns.

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DescribeLedger

Serviço: Amazon QLDB

Retorna informações sobre um ledger, incluindo seu estado, modo de permissões, configurações de criptografia em repouso e quando ele foi criado.

Sintaxe da Solicitação

```
GET /ledgers/name HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger que você deseja descrever.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
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"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Arn

O nome do recurso da Amazon (ARN) para o ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

CreationDateTime

A data e a hora, no formato epoch, quando o ledger foi criado. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Tipo: carimbo de data/hora

DeletionProtection

Especifica se o ledger está protegido contra exclusão por qualquer usuário. Se ele não for fornecido na criação do razão, esse recurso estará habilitado (`true`) por padrão.

Se a proteção contra exclusão estiver habilitada, você deverá desabilitá-la antes de excluir o razão. Você pode desabilitá-lo chamando a operação `UpdateLedger` para definir esse parâmetro como `false`.

Tipo: booliano

EncryptionDescription

Informações sobre a criptografia de dados em repouso em um ledger. Isso inclui o status atual, a AWS KMS chave e quando a chave ficou inacessível (no caso de um erro). Se esse parâmetro for indefinido, o livro contábil usará uma chave AWS KMS própria para criptografia.

Tipo: objeto LedgerEncryptionDescription

Name

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

O modo de permissões atual do ledger.

Tipo: sequências

Valores Válidos: `ALLOW_ALL | STANDARD`

State

O status atual do ledger.

Tipo: sequências

Valores Válidos: `CREATING | ACTIVE | DELETING | DELETED`

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ExportJournalToS3

Serviço: Amazon QLDB

Exporta o conteúdo do diário em um intervalo de data e hora de um ledger para um bucket do Amazon Simple Storage Service (Amazon S3). Um trabalho de exportação de diário pode gravar os objetos de dados no texto ou na representação binária do formato Amazon Ion ou no formato de texto JSON Lines.

Se o livro com o dado `Name` não existir, então lança `ResourceNotFoundException`.

Se o livro com o dado `Name` estiver no status `CREATING`, então lança `ResourcePreconditionNotMetException`.

Você pode iniciar até duas solicitações simultâneas de exportação de diário para cada ledger. Além desse limite, as solicitações de exportação de diário lançam `LimitExceededException`.

Sintaxe da Solicitação

```
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"
   }
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

ExclusiveEndTime

A data e hora de término exclusivas da variedade de conteúdos do periódico a serem exportados.

O `ExclusiveEndTime` deve estar no formato de data e hora `ISO 8601` e em UTC (Tempo Universal Coordenado). Por exemplo: `2019-06-13T21:36:34Z`.

O `ExclusiveEndTime` deve ser menor ou igual à data e hora UTC atuais.

Tipo: carimbo de data/hora

Obrigatório: Sim

InclusiveStartTime

A data e hora de início exclusivas da variedade de conteúdos do periódico a serem exportados.

O `InclusiveStartTime` deve estar no formato de data e hora `ISO 8601` e em UTC (Tempo Universal Coordenado). Por exemplo: `2019-06-13T21:36:34Z`.

O `InclusiveStartTime` deve ser antes de `ExclusiveEndTime`.

Se você fornecer uma `InclusiveStartTime` anterior à `CreationDateTime` do ledger, o Amazon QLDB efetivamente usará como padrão a `CreationDateTime` do ledger.

Tipo: carimbo de data/hora

Obrigatório: Sim

OutputFormat

O formato de saída dos dados exportados do diário. Um trabalho de exportação de diário pode gravar os objetos de dados no texto ou na representação binária do formato Amazon Ion ou no formato de texto JSON Lines.

Padrão: `ION_TEXT`

No formato JSON Lines, cada bloco de diário em um objeto de dados exportado é um objeto JSON válido delimitado por uma nova linha. Você pode usar esse formato para integrar diretamente as exportações JSON com ferramentas de análise, como o Amazon Athena e AWS Glue porque esses serviços podem analisar automaticamente o JSON delimitado por novas linhas.

Tipo: sequências

Valores Válidos: `ION_BINARY | ION_TEXT | JSON`

Obrigatório: não

RoleArn

O Nome do recurso da Amazon (ARN) do perfil do IAM que concede ao QLDB permissões para um trabalho de exportação de diário fazer o seguinte:

- Gravar objetos em um bucket do Amazon S3.
- (Opcional) Use sua chave gerenciada pelo cliente em AWS Key Management Service (AWS KMS) para criptografia do lado do servidor dos dados exportados.

Para transmitir uma função ao QLDB ao solicitar uma exportação de diário, você deve ter permissões para realizar a ação `iam:PassRole` no recurso do perfil do IAM. Isso é necessário para todas as solicitações de exportação de diário.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

S3ExportConfiguration

As configurações do destino do bucket do Amazon S3 para a solicitação de exportação.

Tipo: objeto S3ExportConfiguration

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "ExportId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

ExportId

O UUID (representado em texto codificado em Base62) que o QLDB atribui a cada trabalho de exportação de diário.

Para descrever sua solicitação de exportação e verificar o status do trabalho, você pode usar `ExportId` para chamar `DescribeJournalS3Export`.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Erros

Para obter informações sobre os erros comuns a todas as ações, consulte Erros comuns.

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBlock

Serviço: Amazon QLDB

Retorna um objeto de bloco em um endereço especificado em um diário. Também devolve uma prova do bloco especificado para verificação, se `DigestTipAddress` for fornecida.

Para obter mais informações sobre o conteúdo dos dados em um bloco, consulte o conteúdo do diário no Guia do desenvolvedor do Amazon QLDB.

Se o ledger especificado não existir ou estiver em status`DELETING` , `ResourceNotFoundException` será lançado.

Se o ledger especificado estiver em status`CREATING` , `ResourcePreconditionNotMetException` será lançado.

Se não existir nenhum bloco com o endereço especificado, será lançado `InvalidParameterException`.

Sintaxe da Solicitação

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

BlockAddress

A localização do bloco que você deseja solicitar. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipo: objeto ValueHolder

Obrigatório: Sim

DigestTipAddress

O último local do bloco coberto pelo resumo para o qual solicitar uma prova. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipo: objeto ValueHolder

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
```

```
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Block

O objeto de dados do bloco no formato Amazon Ion.

Tipo: objeto ValueHolder

Proof

O objeto de prova no formato Amazon Ion retornado por uma solicitação `GetBlock`. Uma prova contém a lista de valores de hash necessários para recalcular o resumo especificado usando uma árvore Merkle, começando com o bloco especificado.

Tipo: objeto ValueHolder

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetDigest

Serviço: Amazon QLDB

Retorna o resumo de um ledger no último bloco confirmado no diário. A resposta inclui um valor de hash de 256 bits e um endereço do bloco.

Sintaxe da Solicitação

```
POST /ledgers/name/digest HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Digest

O valor de hash de 256 bits que representa o resumo retornado por uma solicitação. `GetDigest`

Tipo: Objeto de dados binários codificado em Base64

Restrições de comprimento: comprimento fixo de 32.

DigestTipAddress

O último local do bloco coberto pelo resumo que você solicitou. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Tipo: objeto ValueHolder

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetRevision

Serviço: Amazon QLDB

Retorna um objeto de dados de revisão para uma ID do documento e endereço do bloco especificados. Também devolve uma prova da revisão especificada para verificação, se `DigestTipAddress` for fornecida.

Sintaxe da Solicitação

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

BlockAddress

A localização do bloco de diário da revisão do documento a ser verificada. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipo: objeto ValueHolder

Obrigatório: Sim

DigestTipAddress

O último local do bloco coberto pelo resumo para o qual solicitar uma prova. Um endereço é uma estrutura Amazon Ion que tem dois campos: `strandId` e `sequenceNo`.

Por exemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipo: objeto ValueHolder

Obrigatório: Não

DocumentId

O UUID (representado em texto codificado em Base62) do documento a ser verificado.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
   "Revision": {
```

```
        "IonText": "string"
    }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Proof

O objeto de prova no formato Amazon Ion retornado por uma solicitação `GetRevision`. Uma prova contém a lista de valores de hash necessários para recalcular o resumo especificado usando uma árvore Merkle, começando com a revisão do documento especificada.

Tipo: objeto ValueHolder

Revision

O objeto de dados de revisão do documento no formato Amazon Ion.

Tipo: objeto ValueHolder

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListJournalKinesisStreamsForLedger

Serviço: Amazon QLDB

Retorna todos os streams de diário para um determinado ledger do Amazon QLDB.

Essa ação não retorna nenhum stream de diário expirado. Para obter mais informações, consulte Expiração para fluxos terminais no Guia do desenvolvedor do Amazon QLDB.

Essa ação retorna um máximo de itens `MaxResults`. Ele é paginado para que você possa recuperar todos os itens chamando `ListJournalKinesisStreamsForLedger` várias vezes.

Sintaxe da Solicitação

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

MaxResults

O número máximo de resultados a serem retornados em uma única solicitação `ListJournalKinesisStreamsForLedger`. (O número real de resultados retornados pode ser menor.)

Faixa válida: valor mínimo de 1. Valor máximo de 100.

NextToken

Um token de paginação, indicando que você deseja recuperar a próxima página de resultados. Se você recebeu um valor para `NextToken` na resposta de uma chamada `ListJournalKinesisStreamsForLedger` anterior, use esse valor como entrada aqui.

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
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"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

NextToken

- Se `NextToken` estiver vazio, a última página de resultados foi processada e não há mais resultados a serem recuperados.

- Se `NextToken` não estiver vazio, há mais resultados disponíveis. Para recuperar a próxima página de resultados, use o valor de `NextToken` em uma chamada `ListJournalKinesisStreamsForLedger` subsequente.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Streams

Os fluxos de diário do QLDB que estão atualmente associados ao ledger fornecido.

Tipo: matriz de objetos JournalKinesisStreamDescription

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListJournalS3Exports

Serviço: Amazon QLDB

Retorna todos os trabalhos de exportação de diários para todos os ledgers associados ao Conta da AWS atual e à região.

Essa ação retorna um máximo de itens `MaxResults`, e é paginada para que você possa recuperar todos os itens chamando `ListJournalS3Exports` várias vezes.

Essa ação não retorna nenhum trabalho de exportação expirado. Para obter mais informações, consulte Expiração de trabalho de exportação no Guia do desenvolvedor do Amazon QLDB.

Sintaxe da Solicitação

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

MaxResults

O número máximo de resultados a serem retornados em uma única solicitação `ListJournalS3Exports`. (O número real de resultados retornados pode ser menor.)

Faixa válida: valor mínimo de 1. Valor máximo de 100.

NextToken

Um token de paginação, indicando que você deseja recuperar a próxima página de resultados. Se você recebeu um valor para `NextToken` na resposta de uma chamada `ListJournalS3Exports` anterior, use esse valor como entrada aqui.

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
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"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

JournalS3Exports

Os trabalhos de exportação de diários para todos os ledgers que estão associados ao Conta da AWS atual e à região.

Tipo: matriz de objetos JournalS3ExportDescription

NextToken

- Se `NextToken` estiver vazio, a última página de resultados foi processada e não há mais resultados a serem recuperados.
- Se `NextToken` não estiver vazio, há mais resultados disponíveis. Para recuperar a próxima página de resultados, use o valor de `NextToken` em uma chamada `ListJournalS3Exports` subsequente.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Erros

Para obter informações sobre os erros comuns que todas as ações retornam, consulte Erros comuns.

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListJournalS3ExportsForLedger

Serviço: Amazon QLDB

Retorna todos os trabalhos de exportação de diário para um ledger especificado.

Essa ação retorna um máximo de itens `MaxResults` e é paginada para que você possa recuperar todos os itens chamando `ListJournalS3ExportsForLedger` várias vezes.

Essa ação não retorna nenhum trabalho de exportação expirado. Para obter mais informações, consulte Expiração de trabalho de exportação no Guia do desenvolvedor do Amazon QLDB.

Sintaxe da Solicitação

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

MaxResults

O número máximo de resultados a serem retornados em uma única solicitação `ListJournalS3ExportsForLedger`. (O número real de resultados retornados pode ser menor.)

Faixa válida: valor mínimo de 1. Valor máximo de 100.

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

NextToken

Um token de paginação, indicando que você deseja recuperar a próxima página de resultados. Se você recebeu um valor para `NextToken` na resposta de uma chamada `ListJournalS3ExportsForLedger` anterior, use esse valor como entrada aqui.

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
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"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

JournalS3Exports

Retorna todos os trabalhos de exportação de diários atualmente associados ao ledger especificado.

Tipo: matriz de objetos JournalS3ExportDescription

NextToken

- Se `NextToken` estiver vazio, a última página de resultados foi processada e não há mais resultados a serem recuperados.
- Se `NextToken` não estiver vazio, há mais resultados disponíveis. Para recuperar a próxima página de resultados, use o valor de `NextToken` em uma chamada `ListJournalS3ExportsForLedger` subsequente.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Erros

Para obter informações sobre os erros comuns que todas as ações retornam, consulte Erros comuns.

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

ListLedgers

Serviço: Amazon QLDB

Retorna todos os livros contábeis associados à corrente Conta da AWS e à região.

Essa ação retorna um máximo de itens `MaxResults` e é paginada para que você possa recuperar todos os itens chamando `ListLedgers` várias vezes.

Sintaxe da Solicitação

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

MaxResults

O número máximo de resultados a serem retornados em uma única solicitação `ListLedgers`. (O número real de resultados retornados pode ser menor.)

Faixa válida: valor mínimo de 1. Valor máximo de 100.

NextToken

Um token de paginação, indicando que você deseja recuperar a próxima página de resultados. Se você recebeu um valor para `NextToken` na resposta de uma chamada `ListLedgers` anterior, use esse valor como entrada aqui.

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Ledgers": [
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Ledgers

Os livros contábeis associados à corrente Conta da AWS e à região.

Tipo: matriz de objetos LedgerSummary

NextToken

Um token de paginação, indicando se há mais resultados disponíveis:

- Se `NextToken` estiver vazio, a última página de resultados foi processada e não há mais resultados a serem recuperados.
- Se `NextToken` não estiver vazio, há mais resultados disponíveis. Para recuperar a próxima página de resultados, use o valor de `NextToken` em uma chamada `ListLedgers` subsequente.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Erros

Para obter informações sobre os erros comuns que todas as ações retornam, consulte Erros comuns.

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListTagsForResource

Serviço: Amazon QLDB

Retorna todas as tags de um recurso específico do Amazon QLDB.

Sintaxe da Solicitação

```
GET /tags/resourceArn HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

resourceArn

O nome do recurso da Amazon (ARN) para o qual listar as tags. Por exemplo: .

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Tags

Retorna as tags associadas ao recurso especificado do Amazon QLDB.

Tipo: mapa de string para string

Entradas do mapa: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de Tamanho de Chave: Tamanho mínimo de 1. O tamanho máximo é 128.

Restrições de tamanho do valor: tamanho mínimo de 0. Tamanho máximo de 256.

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2

- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

StreamJournalToKinesis

Serviço: Amazon QLDB

Cria um stream de diário para um ledger do Amazon QLDB. O stream captura todas as revisões de documentos confirmadas no diário do razão e entrega os dados a um recurso especificado do Amazon Kinesis Data Streams.

Sintaxe da Solicitação

```
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"
   }
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

ExclusiveEndTime

A data e hora exclusivas que especificam quando o stream termina. Se você não definir esse parâmetro, o stream será executado indefinidamente até ser cancelado.

O `ExclusiveEndTime` deve estar no formato de data e hora `ISO 8601` e em UTC (Tempo Universal Coordenado). Por exemplo: `2019-06-13T21:36:34Z`.

Tipo: carimbo de data/hora

Obrigatório: não

InclusiveStartTime

A data e hora de início inclusivas a partir das quais iniciar o streaming de dados do diário. Esse parâmetro deve estar no formato de data e hora `ISO 8601` e em UTC (Tempo Universal Coordenado). Por exemplo: `2019-06-13T21:36:34Z`.

O `InclusiveStartTime` não pode ser no futuro e deve ser antes de `ExclusiveEndTime`.

Se você fornecer uma `InclusiveStartTime` anterior à `CreationDateTime` do razão, o QLDB efetivamente usará como padrão a `CreationDateTime` do razão.

Tipo: carimbo de data/hora

Obrigatório: Sim

KinesisConfiguration

As configurações do destino do Kinesis Data Streams para a solicitação de stream.

Tipo: objeto KinesisConfiguration

Obrigatório: Sim

RoleArn

O nome do recurso da Amazon (ARN) da perfil do IAM que concede ao QLDB permissões para um stream de diário gravar registros de dados em um recurso do Kinesis Data Streams.

Para transmitir uma função ao QLDB ao solicitar um fluxo de diário, você deve ter permissões para realizar a ação `iam:PassRole` no recurso do perfil do IAM. Isso é necessário para todas as solicitações de fluxos de diário.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

StreamName

O nome que você deseja atribuir ao stream de diário do QLDB. Os nomes definidos pelo usuário podem ajudar a identificar e indicar a finalidade de um stream.

O nome do fluxo deve ser exclusivo entre outros fluxos ativos em um ledger. Os nomes de streams têm as mesmas restrições de nomenclatura que os nomes do razão, conforme definido em Cotas no Amazon QLDB no Guia do desenvolvedor do Amazon QLDB.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Tags

Os pares de chave-valor a serem adicionados como tags ao stream que você deseja criar. Chaves de tag fazem distinção entre maiúsculas e minúsculas. Os valores de tag diferenciam maiúsculas de minúsculas e podem ser nulos.

Tipo: mapa de string para string

Entradas do mapa: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de Tamanho de Chave: Tamanho mínimo de 1. O tamanho máximo é 128.

Restrições de tamanho do valor: tamanho mínimo de 0. Tamanho máximo de 256.

Obrigatório: Não

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "StreamId": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

StreamId

O UUID (representado em texto codificado em Base62) que o QLDB atribui a cada fluxo de diário do QLDB.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

ResourcePreconditionNotMetException

A operação falhou porque uma condição não foi satisfeita com antecedência.

Código de status HTTP: 412

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

TagResource

Serviço: Amazon QLDB

Adiciona uma ou mais tags a um recurso Amazon QLDB especificado.

Um recurso pode ter até 50 tags. Se você tentar criar mais de 50 tags para um recurso, sua solicitação falhará e retornará um erro.

Sintaxe da Solicitação

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

resourceArn

O Nome do recurso da Amazon (ARN) para os quais você deseja adicionar as tags. Por exemplo: .

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

Tags

Os pares de chave-valor a serem adicionados como tags ao recurso QLDB especificado. Chaves de tag fazem distinção entre maiúsculas e minúsculas. Se você especificar uma chave que já

existe para o recurso, sua solicitação falhará e retornará um erro. Os valores de tag diferenciam maiúsculas de minúsculas e podem ser nulos.

Tipo: mapa de string para string

Entradas do mapa: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de Tamanho de Chave: Tamanho mínimo de 1. O tamanho máximo é 128.

Restrições de tamanho do valor: tamanho mínimo de 0. Tamanho máximo de 256.

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 200
```

Elementos de Resposta

Se a ação tiver êxito, o serviço enviará de volta uma resposta HTTP 200 com um corpo HTTP vazio.

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UntagResource

Serviço: Amazon QLDB

Remove uma ou mais tags de um recurso Amazon QLDB especificado. Você pode especificar até 50 chaves de tags para remover.

Sintaxe da Solicitação

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

resourceArn

O nome do recurso da Amazon (ARN) do recurso do qual remover as tags. Por exemplo: .

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

TagKeys

A lista de chaves de tags a serem removidas.

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: Sim

Corpo da Solicitação

Essa solicitação não tem corpo.

Sintaxe da Resposta

```
HTTP/1.1 200
```

Elementos de Resposta

Se a ação tiver êxito, o serviço enviará de volta uma resposta HTTP 200 com um corpo HTTP vazio.

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UpdateLedger

Serviço: Amazon QLDB

Atualiza propriedades em um ledger.

Sintaxe da Solicitação

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

DeletionProtection

Especifica se o ledger está protegido contra exclusão por qualquer usuário. Se ele não for fornecido na criação do razão, esse recurso estará habilitado (`true`) por padrão.

Se a proteção contra exclusão estiver habilitada, você deverá desabilitá-la antes de excluir o razão. Você pode desabilitá-lo chamando a operação `UpdateLedger` para definir esse parâmetro como `false`.

Tipo: booliano

Obrigatório: não

KmsKey

A chave in AWS Key Management Service (AWS KMS) a ser usada para criptografia de dados em repouso no livro contábil. Para obter mais informações, consulte Criptografia em repouso, no Guia do desenvolvedor do Amazon QLDB.

Utilize uma das seguintes opções para especificar esse parâmetro:

- `AWS_OWNED_KMS_KEY`: use uma AWS KMS chave que pertença e seja AWS gerenciada por você.
- Indefinido: não faz alterações na chave KMS do ledger.
- Uma chave simétrica do KMS válida e gerenciada pelo cliente: use a chave do KMS de criptografia simétrica, que você cria, detém e gerencia, especificada na sua conta.

  O Amazon QLDB não oferece suporte a chaves assimétricas. Para obter mais informações, consulte Como usar chaves simétricas e assimétricas no Guia do AWS Key Management Service desenvolvedor.

Para especificar uma chave do KMS gerenciada pelo cliente, use o ID da chave, o nome do recurso da Amazon (ARN), o nome do alias ou o ARN do alias. Ao usar um nome de alias, use `"alias/"` como prefixo. Para especificar uma chave em outra Conta da AWS, você deve usar o ARN da chave ou o alias ARN.

Por exemplo: .

- ID da chave: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN da chave: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nome do alias: `alias/ExampleAlias`
- ARN do alias: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Para obter mais informações, consulte Identificadores de chave (KeyId) no Guia do AWS Key Management Service desenvolvedor.

Tipo: sequência

Restrições de tamanho: tamanho máximo de 1600.

Obrigatório: Não

Sintaxe da Resposta

```
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"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Arn

O nome do recurso da Amazon (ARN) para o ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

CreationDateTime

A data e a hora, no formato epoch, quando o ledger foi criado. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Tipo: carimbo de data/hora

DeletionProtection

Especifica se o ledger está protegido contra exclusão por qualquer usuário. Se ele não for fornecido na criação do razão, esse recurso estará habilitado (`true`) por padrão.

Se a proteção contra exclusão estiver habilitada, você deverá desabilitá-la antes de excluir o razão. Você pode desabilitá-lo chamando a operação `UpdateLedger` para definir esse parâmetro como `false`.

Tipo: booliano

EncryptionDescription

Informações sobre a criptografia de dados em repouso em um ledger. Isso inclui o status atual, a AWS KMS chave e quando a chave ficou inacessível (no caso de um erro).

Tipo: objeto LedgerEncryptionDescription

Name

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

O status atual do ledger.

Tipo: sequências

Valores Válidos: `CREATING | ACTIVE | DELETING | DELETED`

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UpdateLedgerPermissionsMode

Serviço: Amazon QLDB

Atualizar o modo de permissões de um ledger.

> **Important**
>
> Antes de mudar para o modo de permissões `STANDARD`, você deve primeiro criar todas as políticas do IAM e tags de tabela necessárias para evitar interrupções para seus usuários. Para saber mais, consulte Migração para o modo de permissões padrão no Guia do desenvolvedor do Amazon QLDB.

Sintaxe da Solicitação

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

Parâmetros da Solicitação de URI

A solicitação usa os seguintes parâmetros de URI:

name

O nome do ledger.

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Corpo da Solicitação

A solicitação aceita os dados a seguir no formato JSON.

PermissionsMode

O modo de permissões a ser atribuído ao ledger. Esse parâmetro pode ter um dos valores a seguir:

- `ALLOW_ALL`: um modo de permissões legado que permite o controle do acesso com granularidade em nível de API para razões.

  Esse modo permite aos usuários que tenham a permissão de API `SendCommand` para esse ledger executar todos os comandos PartiQL (portanto, `ALLOW_ALL`) em qualquer tabela no razão especificado. Esse modo desconsidera qualquer política de permissões do IAM em nível de tabela ou comando criada para o razão.
- `STANDARD`: (Recomendado) Um modo de permissões que permite o controle do acesso com granularidade mais fina para razões, tabelas e comandos PartiQL.

  Por padrão, esse modo nega todas as solicitações do usuário para executar comandos do PartiQL em qualquer tabela nesse razão. Para permitir a execução de comandos PartiQL, é necessário criar políticas de permissões do IAM para recursos de tabela específicos e ações PartiQL, além da permissão da API `SendCommand` para o razão. Para saber mais, consulte Conceitos básicos sobre o modo de permissões padrão, no Guia do desenvolvedor do Amazon QLDB.

> **Note**
>
> Recomendamos o uso do modo de permissões `STANDARD` para maximizar a segurança dos dados do seu razão.

Tipo: sequências

Valores Válidos: `ALLOW_ALL | STANDARD`

Exigido: Sim

Sintaxe da Resposta

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "Arn": "string",
    "Name": "string",
    "PermissionsMode": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

Arn

O nome do recurso da Amazon (ARN) para o ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Name

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

O modo de permissões atual do ledger.

Tipo: sequências

Valores Válidos: `ALLOW_ALL | STANDARD`

Erros

Para obter informações sobre os erros comuns que são comuns a todas as ações, consulte Erros comuns.

InvalidParameterException

Um ou mais parâmetros na solicitação não são válidos.

Código de Status HTTP: 400

ResourceNotFoundException

O recurso especificado não existe.

Código de Status HTTP: 404

Consulte Também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

QLDBSessão Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

As seguintes ações são apoiadas pela Amazon QLDB Session:

- SendCommand

SendCommand

Serviço: Amazon QLDB Session

Envia um comando para um ledger do Amazon QLDB.

> **Note**
>
> Em vez de interagir diretamente com essa API, recomendamos usar o driver QLDB ou o shell QLDB para executar transações de dados em um ledger.
>
> - Se você estiver trabalhando com um AWS SDK, use o driver QLDB. O driver fornece uma camada de abstração de alto nível acima dessa API de dados do QLDB Session e gerencia a operação `SendCommand` para você. Para obter informações e uma lista das linguagens de programação suportadas, consulte Conceitos básicos do driver no Guia do usuário Amazon QLDB.
> - Se você estiver trabalhando com o AWS Command Line Interface (AWS CLI), use o shell QLDB. O shell é uma interface de linha de comando que usa o driver QLDB para interagir com um ledger. Para obter informações, consulte Acessando o Amazon QLDB usando o shell QLDB.

Sintaxe da Solicitação

```
{
   "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": {
    }
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros que são comuns em todas as ações, consulte Parâmetros comuns.

A solicitação aceita os dados a seguir no formato JSON.

AbortTransaction

Comando para abortar a transação atual.

Tipo: objeto AbortTransactionRequest

Obrigatório: Não

CommitTransaction

Comando para confirmar a transação especificada.

Tipo: objeto CommitTransactionRequest

Obrigatório: Não

EndSession

Comando para encerrar a sessão atual.

Tipo: objeto EndSessionRequest

Obrigatório: Não

ExecuteStatement

Comando para executar uma instrução na transação especificada.

Tipo: objeto ExecuteStatementRequest

Obrigatório: Não

FetchPage

Comando para buscar uma página.

Tipo: objeto FetchPageRequest

Obrigatório: Não

SessionToken

Especifica o token da sessão para o comando atual. Um token de sessão é constante durante toda a duração da sessão.

Para obter um token de sessão, execute o comando `StartSession`. Esse `SessionToken` é necessário para cada comando subsequente emitido durante a sessão atual.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Obrigatório: não

StartSession

Comando para iniciar uma nova sessão. Um token de sessão é obtido como parte da resposta.

Tipo: objeto StartSessionRequest

Obrigatório: Não

StartTransaction

Comando para iniciar uma nova transação.

Tipo: objeto StartTransactionRequest

Obrigatório: Não

Sintaxe da Resposta

```
{
   "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"
    }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP 200.

Os dados a seguir são retornados no formato JSON pelo serviço.

AbortTransaction

Contém os detalhes da transação cancelada.

Tipo: objeto AbortTransactionResult

CommitTransaction

Contém os detalhes da transação confirmada.

Tipo: objeto CommitTransactionResult

EndSession

Contém os detalhes da sessão encerrada.

Tipo: objeto EndSessionResult

ExecuteStatement

Contém os detalhes da instrução executada.

Tipo: objeto ExecuteStatementResult

FetchPage

Contém os detalhes da página buscada.

Tipo: objeto FetchPageResult

StartSession

Contém os detalhes da sessão iniciada que inclui um token de sessão. Esse `SessionToken` é necessário para cada comando subsequente emitido durante a sessão atual.

Tipo: objeto StartSessionResult

StartTransaction

Contém os detalhes da transação iniciada.

Tipo: objeto StartTransactionResult

Erros

Para obter informações sobre os erros comuns a todas as ações, consulte Erros comuns.

BadRequestException

Retornado se a solicitação estiver malformada ou contiver um erro, como um valor de parâmetro inválido ou um parâmetro obrigatório ausente.

Código de Status HTTP: 400

CapacityExceededException

Retornado quando a solicitação excede a capacidade de processamento do ledger.

Código de Status HTTP: 400

InvalidSessionException

Retornado se a sessão não existir mais porque ultrapassou o limite de tempo ou expirou.

Código de Status HTTP: 400

LimitExceededException

Retornado se um limite de recursos, como o número de sessões ativas, for excedido.

Código de Status HTTP: 400

OccConflictException

Retornado quando uma transação não pode ser gravada no diário devido a uma falha na fase de verificação do controle otimista de simultaneidade (OCC).

Código de Status HTTP: 400

RateExceededException

Retornado quando a taxa de solicitações excede o throughput permitido.

Código de Status HTTP: 400

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

Tipos de dados

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Os seguintes tipos de dados são compatíveis com a AmazonQLDB:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

Os seguintes tipos de dados são compatíveis com a Amazon QLDB Session:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage

- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Os seguintes tipos de dados são compatíveis com a AmazonQLDB:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

JournalKinesisStreamDescription

Serviço: Amazon QLDB

Informações sobre um stream de diário do Amazon QLDB, incluindo o nome do recurso da Amazon (ARN), nome do stream, horário de criação, status atual e os parâmetros da solicitação original de criação do stream.

Conteúdo

KinesisConfiguration

As configurações do destino do Amazon Kinesis Data Streams para um stream de diário do QLDB.

Tipo: objeto KinesisConfiguration

Obrigatório: Sim

LedgerName

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

RoleArn

O nome do recurso da Amazon (ARN) da perfil do IAM que concede ao QLDB permissões para um stream de diário gravar registros de dados em um recurso do Kinesis Data Streams.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

Status

O estado atual do stream de diário do QLDB.

Tipo: sequências

Valores Válidos: `ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED`

Obrigatório: Sim

StreamId

O UUID (representado em texto codificado em Base62) do stream de diário do QLDB.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

StreamName

O nome definido pelo usuário do stream de diário do QLDB.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Arn

O nome de recurso da Amazon (ARN) do stream de diário do QLDB.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: não

CreationTime

A data e a hora, em formato de epoch time, quando o stream de diário do QLDB foi criado. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Tipo: carimbo de data/hora

Obrigatório: não

ErrorCause

A mensagem de erro que descreve o motivo pelo qual um stream tem um status de `IMPAIRED` ou`FAILED`. Isso não se aplica a streams que tenham outros valores de status.

Tipo: sequências

Valores Válidos: `KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED`

Obrigatório: não

ExclusiveEndTime

A data e hora exclusivas que especificam quando o stream termina. Se você não definir esse parâmetro, o stream será executado indefinidamente até ser cancelado.

Tipo: carimbo de data/hora

Obrigatório: não

InclusiveStartTime

A data e hora de início inclusivas a partir das quais iniciar o streaming de dados do diário.

Tipo: carimbo de data/hora

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

JournalS3ExportDescription

Serviço: Amazon QLDB

Retorna informações sobre um trabalho de exportação de diário, incluindo o nome do ledger, a ID de exportação, a hora da criação, o status atual e os parâmetros da solicitação original de criação da exportação.

Conteúdo

ExclusiveEndTime

A data e hora de término exclusivas da variedade de conteúdos do periódico especificados na solicitação original de exportação.

Tipo: carimbo de data/hora

Obrigatório: Sim

ExportCreationTime

A data e a hora, no formato epoch, quando o trabalho de exportação foi criado. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Tipo: carimbo de data/hora

Obrigatório: Sim

ExportId

O UUID (representado em texto codificado em Base62) do trabalho de exportação de diário.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

InclusiveStartTime

A data e hora de término exclusivas da variedade de conteúdos do periódico especificados na solicitação original de exportação.

Tipo: carimbo de data/hora

Obrigatório: Sim

LedgerName

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

RoleArn

O Nome do recurso da Amazon (ARN) do perfil do IAM que concede ao QLDB permissões para um trabalho de exportação de diário fazer o seguinte:

- Gravar objetos em seu bucket do Amazon Simple Storage Service (Amazon S3).
- (Opcional) Use sua chave gerenciada pelo cliente em AWS Key Management Service (AWS KMS) para criptografia do lado do servidor dos dados exportados.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

S3ExportConfiguration

O local do bucket do Amazon Simple Storage Service (Amazon S3) em que um trabalho de exportação de diário grava o conteúdo do diário.

Tipo: objeto S3ExportConfiguration

Obrigatório: Sim

Status

O estado atual do trabalho de exportação do diário.

Tipo: sequências

Valores Válidos: `IN_PROGRESS | COMPLETED | CANCELLED`

Obrigatório: Sim

OutputFormat

O formato de saída dos dados exportados do diário.

Tipo: sequências

Valores Válidos: `ION_BINARY | ION_TEXT | JSON`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

KinesisConfiguration

Serviço: Amazon QLDB

As configurações do destino do Amazon Kinesis Data Streams para um stream de diário do Amazon QLDB.

Conteúdo

StreamArn

Anote o nome de recurso da Amazon (ARN) do recurso do Kinesis Data Streams.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

AggregationEnabled

Permite que o QLDB publique vários registros de dados em um único registro do Kinesis Data Streams, aumentando o número de registros enviados por chamada de API.

Padrão: `True`

> **Important**
>
> A agregação de registros tem implicações importantes para o processamento de registros e requer desagregação em seu consumidor de stream. Para saber mais, consulte Conceitos-chave de KPL e Desagregação do consumidor no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Tipo: booliano

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LedgerEncryptionDescription

Serviço: Amazon QLDB

Informações sobre a criptografia de dados em repouso em um ledger do Amazon QLDB. Isso inclui o status atual, a chave em AWS Key Management Service (AWS KMS) e quando a chave ficou inacessível (no caso de um erro).

Para obter mais informações, consulte Criptografia em repouso, no Guia do desenvolvedor do Amazon QLDB.

Conteúdo

EncryptionStatus

O status atual da criptografia em repouso para o ledger. Pode ter um dos valores a seguir:

- `ENABLED`: A criptografia é totalmente ativada usando a chave especificada.
- `UPDATING`: O livro contábil está processando ativamente a alteração de chave especificada.

  As alterações de chave no QLDB são assíncronas. O ledger é totalmente acessível sem nenhum impacto no desempenho enquanto a alteração da chave está sendo processada. O tempo necessário para atualizar uma chave varia dependendo do tamanho do ledger.
- `KMS_KEY_INACCESSIBLE`: A chave KMS gerenciada pelo cliente especificada não está acessível e o ledger está danificado. Ou a chave foi desativada ou excluída, ou as concessões da chave foram revogadas. Quando um ledger está danificado, ele não está acessível e não aceita nenhuma solicitação de leitura ou gravação.

  Um ledger danificado retorna automaticamente ao estado ativo depois que você restaura as concessões na chave ou depois de reativar a chave que foi desativada. Entretanto, a exclusão de uma chave KMS gerenciada pelo cliente é irreversível. Depois que uma chave é excluída, não é mais possível acessar os ledgers que estão protegidos com ela, e os dados ficam irrecuperáveis permanentemente

Tipo: sequências

Valores Válidos: `ENABLED | UPDATING | KMS_KEY_INACCESSIBLE`

Obrigatório: Sim

KmsKeyArn

O nome do recurso da Amazon (ARN) da chave KMS gerenciada pelo cliente que o ledger usa para criptografia em repouso. Se esse parâmetro for indefinido, o livro contábil usará uma chave AWS KMS própria para criptografia. Ele será exibido `AWS_OWNED_KMS_KEY` ao atualizar a configuração de criptografia do livro contábil para a chave KMS AWS de propriedade.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: Sim

InaccessibleKmsKeyDateTime

A data e a hora, no formato de época, em que a AWS KMS chave ficou inacessível pela primeira vez, no caso de um erro. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Esse parâmetro é indefinido se a AWS KMS chave estiver acessível.

Tipo: carimbo de data/hora

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LedgerSummary

Serviço: Amazon QLDB

Informações sobre um ledger, incluindo seu nome, estado e quando ele foi criado.

Conteúdo

CreationDateTime

A data e a hora, no formato epoch, quando o ledger foi criado. (O formato de hora epoch é o número de segundos decorridos desde as 0h de 1 de janeiro de 1970 em UTC.)

Tipo: carimbo de data/hora

Obrigatório: não

Name

O nome do ledger.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obrigatório: não

State

O status atual do ledger.

Tipo: sequências

Valores Válidos: `CREATING | ACTIVE | DELETING | DELETED`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

S3EncryptionConfiguration

Serviço: Amazon QLDB

As configurações de criptografia usadas por um trabalho de exportação de diário para gravar dados em um bucket do Amazon Simple Storage Service (Amazon S3).

Conteúdo

ObjectEncryptionType

O tipo de criptografia do objeto Amazon S3.

Para obter mais informações sobre o uso da criptografia no lado do servidor no Amazon S3, consulte Proteger dados usando criptografia no lado do servidor, no Guia do desenvolvedor do Amazon S3.

Tipo: sequências

Valores Válidos: `SSE_KMS | SSE_S3 | NO_ENCRYPTION`

Obrigatório: Sim

KmsKeyArn

O Amazon Resource Name (ARN) de uma chave de criptografia simétrica em AWS Key Management Service ().AWS KMS O Amazon S3 não oferece suporte a Chaves do KMS assimétricas.

Você deve fornecer um `KmsKeyArn` se especificar `SSE_KMS` como o `ObjectEncryptionType`.

Se você especificar `SSE_S3` como o `ObjectEncryptionType`, `KmsKeyArn` não será obrigatório.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 20. Comprimento máximo de 1.600.

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

S3ExportConfiguration

Serviço: Amazon QLDB

O local do bucket do Amazon Simple Storage Service (Amazon S3) em que um trabalho de exportação de diário grava o conteúdo do diário.

Conteúdo

Bucket

O local do bucket do Amazon Simple Storage Service (Amazon S3) em que um trabalho de exportação de diário grava o conteúdo do diário.

O nome do bucket deve estar de acordo com as convenções de nomenclatura do Amazon S3. Para obter mais informações, consulte Bucket Restrictions and Limitations no Guia do desenvolvedor do Amazon S3.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 3. Comprimento máximo de 255.

Padrão: `^[A-Za-z-0-9-_.]+$`

Exigido: Sim

EncryptionConfiguration

As configurações de criptografia usadas por um trabalho de exportação de diário para gravar dados em um bucket do Amazon S3.

Tipo: objeto S3EncryptionConfiguration

Obrigatório: Sim

Prefix

O prefixo para o bucket do Amazon S3 em que um trabalho de exportação de diário grava o conteúdo do diário.

O prefixo deve estar em conformidade com as regras e restrições de nomenclatura de chaves do Amazon S3. Para obter mais informações sobre nomes de chaves, consulte Chave e metadados de objeto no Guia do desenvolvedor do Amazon S3.

Por exemplo, os seguintes valores `Prefix` são válidos:

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

Tipo: sequência

Restrições de tamanho: tamanho mínimo 0. O tamanho máximo é 128.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ValueHolder

Serviço: Amazon QLDB

Uma estrutura que pode conter um valor em vários formatos de codificação.

Conteúdo

IonText

Um valor de texto simples do Amazon Ion contido em uma estrutura `ValueHolder`.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1048576.

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

QLDBSessão Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Os seguintes tipos de dados são compatíveis com a Amazon QLDB Session:

- AbortTransactionRequest
- AbortTransactionResult

- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

Serviço: Amazon QLDB Session

Contém os detalhes da transação cancelada.

Conteúdo

Os membros dessa estrutura de exceção são dependentes do contexto.

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

AbortTransactionResult

Serviço: Amazon QLDB Session

Contém os detalhes da transação cancelada.

Conteúdo

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

CommitTransactionRequest

Serviço: Amazon QLDB Session

Contém os detalhes da transação a confirmar.

Conteúdo

CommitDigest

Especifica o resumo de confirmação da transação a confirmar. Para cada transação ativa, o resumo de confirmação deve ser passado. O QLDB valida `CommitDigest` e rejeita a confirmação com um erro se o resumo calculado no cliente não corresponder ao resumo calculado pelo QLDB.

O objetivo do parâmetro `CommitDigest` é garantir que o QLDB confirme uma transação se e somente se o servidor tiver processado o conjunto exato de instruções enviadas pelo cliente, na mesma ordem em que o cliente as enviou e sem duplicatas.

Tipo: Objeto de dados binários codificado em Base64

Obrigatório: Sim

TransactionId

Especifica o ID da transação a confirmar.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

CommitTransactionResult

Serviço: Amazon QLDB Session

Contém os detalhes da transação confirmada.

Conteúdo

CommitDigest

O resumo da confirmação da transação confirmada.

Tipo: Objeto de dados binários codificado em Base64

Obrigatório: não

ConsumedIOs

Contém métricas sobre o número de solicitações de E/S que foram consumidas.

Tipo: objeto IOUsage

Obrigatório: Não

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

TransactionId

O ID da transação confirmada.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

EndSessionRequest

Serviço: Amazon QLDB Session

Especifica uma solicitação para encerrar a sessão.

Conteúdo

Os membros dessa estrutura de exceção são dependentes do contexto.

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

EndSessionResult

Serviço: Amazon QLDB Session

Contém os detalhes da sessão encerrada.

Conteúdo

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ExecuteStatementRequest

Serviço: Amazon QLDB Session

Especifica uma solicitação para executar uma instrução.

Conteúdo

Statement

Especifica a instrução da solicitação.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 100000.

Obrigatório: Sim

TransactionId

Especifica o ID da transação da solicitação.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

Parameters

Especifica os parâmetros da instrução parametrizada na solicitação.

Tipo: matriz de objetos ValueHolder

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2

- AWS SDK para Ruby V3

ExecuteStatementResult

Serviço: Amazon QLDB Session

Contém os detalhes da instrução executada.

Conteúdo

ConsumedIOs

Contém métricas sobre o número de solicitações de E/S que foram consumidas.

Tipo: objeto IOUsage

Obrigatório: Não

FirstPage

Contém detalhes da primeira página buscada.

Tipo: objeto Page

Obrigatório: Não

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

FetchPageRequest

Serviço: Amazon QLDB Session

Especifica os detalhes da página a ser buscada.

Conteúdo

NextPageToken

Especifica o próximo token da página a ser buscada.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Exigido: Sim

TransactionId

Especifica os detalhes de ID da transação da página a ser buscada.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

FetchPageResult

Serviço: Amazon QLDB Session

Contém a página que foi buscada.

Conteúdo

ConsumedIOs

Contém métricas sobre o número de solicitações de E/S que foram consumidas.

Tipo: objeto IOUsage

Obrigatório: Não

Page

Contém detalhes da página buscada.

Tipo: objeto Page

Obrigatório: Não

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IOUsage

Serviço: Amazon QLDB Session

Contém métricas de uso de E/S para um comando que foi invocado.

Conteúdo

ReadIOs

O número de solicitações de E/S de leitura executadas pelo comando.

Tipo: longo

Obrigatório: não

WriteIOs

O número de solicitações de E/S de gravação executadas pelo comando.

Tipo: longo

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Page

Serviço: Amazon QLDB Session

Contém detalhes da página buscada.

Conteúdo

NextPageToken

O token da próxima página.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Obrigatório: não

Values

Uma estrutura que pode conter um valor em vários formatos de codificação.

Tipo: matriz de objetos ValueHolder

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartSessionRequest

Serviço: Amazon QLDB Session

Especifica uma solicitação para iniciar uma nova sessão.

Conteúdo

LedgerName

O nome do ledger com o qual iniciar uma nova sessão.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 32.

Padrão: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartSessionResult

Serviço: Amazon QLDB Session

Contém os detalhes da sessão iniciada.

Conteúdo

SessionToken

Token de sessão da sessão iniciada. Esse `SessionToken` é necessário para cada comando subsequente emitido durante a sessão atual.

Tipo: sequência

Restrições de tamanho: tamanho mínimo de 4. Tamanho máximo de 1.024.

Padrão: `^[A-Za-z-0-9+/=]+$`

Obrigatório: não

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartTransactionRequest

Serviço: Amazon QLDB Session

Especifica uma solicitação para iniciar uma transação.

Conteúdo

Os membros dessa estrutura de exceção são dependentes do contexto.

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartTransactionResult

Serviço: Amazon QLDB Session

Contém os detalhes da transação iniciada.

Conteúdo

TimingInformation

Contém informações de desempenho do lado do servidor para o comando.

Tipo: objeto TimingInformation

Obrigatório: Não

TransactionId

O ID da transação iniciada.

Tipo: sequência

Restrições de comprimento: comprimento fixo de 22.

Padrão: `^[A-Za-z-0-9]+$`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

TimingInformation

Serviço: Amazon QLDB Session

Contém informações de desempenho do lado do servidor para um comando. O Amazon QLDB captura informações de tempo entre o momento em que recebe a solicitação e o momento em que envia a resposta correspondente.

Conteúdo

ProcessingTimeMilliseconds

A quantidade de tempo que o QLDB gastou no processamento do comando, medida em milissegundos.

Tipo: longo

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ValueHolder

Serviço: Amazon QLDB Session

Uma estrutura que pode conter um valor em vários formatos de codificação.

Conteúdo

IonBinary

Um valor binários do Amazon Ion contido em uma estrutura `ValueHolder`.

Tipo: Objeto de dados binários codificado pelo Base64

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 131072.

Obrigatório: não

IonText

Um valor de texto simples do Amazon Ion contido em uma estrutura `ValueHolder`.

Tipo: sequência

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1048576.

Obrigatório: não

Consulte também

Para obter mais informações sobre como usar essa API em um dos AWS SDKs específicos da linguagem, consulte o seguinte:

- AWS SDK for C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Erros comuns

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção lista os erros comuns às API ações de todos os AWS serviços. Para erros específicos de uma API ação desse serviço, consulte o tópico dessa API ação.

AccessDeniedException

Você não tem acesso suficiente para executar essa ação.

HTTPCódigo de status: 400

IncompleteSignature

A assinatura da solicitação não está em conformidade com os AWS padrões.

HTTPCódigo de status: 400

InternalFailure

O processamento da solicitação falhou por causa de um erro, uma exceção ou uma falha desconhecida.

HTTPCódigo de status: 500

InvalidAction

A ação ou operação solicitada é inválida. Verifique se a ação foi digitada corretamente.

HTTPCódigo de status: 400

InvalidClientTokenId

O certificado X.509 ou ID da chave de AWS acesso fornecido não existe em nossos registros.

HTTPCódigo de status: 403

NotAuthorized

Você não tem permissão para realizar esta ação.

HTTPCódigo de status: 400

OptInRequired

O ID da chave de AWS acesso precisa de uma assinatura para o serviço.

HTTPCódigo de status: 403

RequestExpired

A solicitação chegou ao serviço mais de 15 minutos após o carimbo de data na solicitação ou mais de 15 minutos após a data de expiração da solicitação (como para pré-assinadaURLs), ou o carimbo de data na solicitação é mais de 15 minutos no futuro.

HTTPCódigo de status: 400

ServiceUnavailable

Falha na solicitação devido a um erro temporário do servidor.

HTTPCódigo de status: 503

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationError

A entrada não satisfaz as restrições especificadas por um AWS serviço.

HTTPCódigo de status: 400

Parâmetros gerais

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

A lista a seguir contém os parâmetros que todas as ações usam para assinar solicitações do Signature versão 4 com uma string de consulta. Todos os parâmetros específicos de uma ação são

listados no tópico para a ação. Para obter mais informações sobre o Signature versão 4, consulte AWS APISolicitações de assinatura no Guia IAM do usuário.

Action

A ação a ser executada.

Tipo: string

Obrigatório: Sim

Version

A API versão para a qual a solicitação foi escrita, expressa no formato YYYY -MM-DD.

Tipo: string

Obrigatório: Sim

X-Amz-Algorithm

O algoritmo de hash que foi usado para criar a assinatura da solicitação.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Valores Válidos: `AWS4-HMAC-SHA256`

Obrigatório: Condicional

X-Amz-Credential

O valor de escopo da credencial, uma string que inclui a sua chave de acesso, a data, a região visada, o serviço que está sendo solicitado e uma sequência de encerramento ("aws4_request"). O valor é expresso no seguinte formato: access_key//region YYYYMMDD/service /aws4_request.

Para obter mais informações, consulte Criar uma AWS API solicitação assinada no Guia IAM do usuário.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Obrigatório: Condicional

X-Amz-Date

A data usada para criar a assinatura. O formato deve ser o formato básico ISO 8601 (YYYYMMDD'T' 'ZHHMMSS'). Por exemplo, a data/hora a seguir é um valor X-Amz-Date válido: `20120325T120000Z`.

Condição: X-Amz-Date é opcional para todas as solicitações e pode ser usado para substituir a data usada para assinar solicitações. Se o cabeçalho da data for especificado no formato básico ISO 8601, o X-Amz-Date não será necessário. Quando X-Amz-Date é usado, sempre substitui o valor do cabeçalho Date (Data). Para obter mais informações, consulte Elementos de uma assinatura de AWS API solicitação no Guia IAM do usuário.

Tipo: string

Obrigatório: Condicional

X-Amz-Security-Token

O token de segurança temporário obtido por meio de uma chamada para AWS Security Token Service (AWS STS). Para obter uma lista de serviços que oferecem suporte a credenciais de segurança temporárias de AWS STS, consulte Serviços da AWS "Trabalhe com" IAM no Guia do IAM usuário.

Condição: se você estiver usando credenciais de segurança temporárias do AWS STS, deverá incluir o token de segurança.

Tipo: string

Obrigatório: Condicional

X-Amz-Signature

Especifica a assinatura com codificação hexadecimal que foi calculada com base na string a ser assinada e na chave de assinatura derivada.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Obrigatório: Condicional

X-Amz-SignedHeaders

Especifica todos os HTTP cabeçalhos que foram incluídos como parte da solicitação canônica. Para obter mais informações sobre a especificação de cabeçalhos assinados, consulte Criar uma AWS API solicitação assinada no Guia do IAMusuário.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Obrigatório: Condicional

Cotas e limites na Amazon QLDB

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Esta seção descreve as cotas atuais, também chamadas de limites, na AmazonQLDB.

Tópicos

- Cotas padrão
- Cotas fixas
- Cota do ledger
- Tamanho do documento
- Tamanho da transação
- Restrições de nomenclatura

Cotas padrão

QLDBtem as seguintes cotas padrão, conforme também listadas nos QLDBendpoints da Amazon e cotas no. Referência geral da AWS Essas cotas são Conta da AWS por região. Para solicitar aumento da cota para sua conta em uma Região, use o console do Service Quotas.

Faça login AWS Management Console e abra o console Service Quotas em. https://console.aws.amazon.com/servicequotas/

Recurso	Cota padrão
O número máximo de ledgers ativos que é possível criar nesta conta na região atual.	5
O número máximo de exportações de diários ativos para o Amazon S3 por ledger	2

Recurso	Cota padrão
O número máximo de fluxos de diário ativos para o Kinesis Data Streams por ledger	5

Cotas fixas

Além das cotas padrão, QLDB tem as seguintes cotas fixas por livro contábil. Essas cotas não podem ser aumentadas usando Service Quotas:

Recurso	Cota fixa
Número de sessões ativas simultâneas	3000
Número de tabelas ativas	20
`IndexedStorage` — quantidade total de espaço em disco usada pelas tabelas, índices e histórico indexado do livro contábil, relatada em intervalos de 15 minutos. Para obter mais informações sobre as CloudWatch métricas da Amazon, consulte Monitoramento com a Amazon CloudWatch.	Máximo de 16 TiB
Número total de tabelas (ativas e inativas) **Note** EmQLDB, as tabelas eliminadas são consideradas inativas e contam para essa cota total.	40
Número de índices por tabela	5
Número de documentos em uma transação	40

Recurso	Cota fixa
Número de revisões a serem redigidas em uma transação	1
Tamanho do documento (codificado em formato `IonBinary`)	128 KB
Tamanho do parâmetro da instrução (formato `IonBinary`)	128 KB
Tamanho do parâmetro da instrução (formato `IonText`)	1 MB
Comprimento da string de instrução	100.000 caracteres
Tamanho da transação	4 MB
Tempo limite de transação	30 segundos
Período de expiração para trabalhos de exportação de diário concluídos	7 dias
Período de expiração para fluxos de diários do terminal	7 dias

Cota do ledger

Para solicitar aumento da cota de ledger para sua conta em uma Região, use o console do Service Quotas.

Abra o console Service Quotas em. https://console.aws.amazon.com/servicequotas/

Alguns casos de QLDB uso exigem um número crescente de livros contábeis Conta da AWS por região com base no crescimento dos negócios. Por exemplo, talvez você precise criar ledgers dedicados para isolar clientes ou dados. Nesse caso, considere usar uma arquitetura de várias contas para trabalhar com QLDB cotas. Para obter mais informações, consulte Account Silo Isolation no AWS whitepaper SaaS Tenant Isolation Strategies.

Tamanho do documento

O tamanho máximo de um documento codificado no formato `IonBinary` é 128 KB. Não podemos fornecer um limite exato para o tamanho de um documento em formato `IonText` porque a conversão de texto para binário varia significativamente com base na estrutura de cada documento. QLDBsuporta documentos com conteúdo aberto, portanto, cada estrutura exclusiva do documento altera o cálculo do tamanho.

Tamanho da transação

O tamanho máximo para uma transação QLDB é de 4 MB. O tamanho de uma transação é calculado com base na soma dos seguintes fatores.

Deltas

As alterações do documento são geradas por todas as instruções na transação. Em uma transação que afeta vários documentos, o tamanho total do delta é a soma do delta individual de cada documento afetado.

Metadados

Os metadados da transação gerados pelo sistema que estão associados a cada documento afetado.

Índices

Se um índice for definido em uma tabela afetada pela transação, a entrada de índice associada também gerará um delta.

Histórico

Como todas as revisões de documentos persistemQLDB, todas as transações também são anexadas ao histórico.

Inserções: cada documento inserido em uma tabela também tem uma cópia inserida em sua tabela de histórico. Por exemplo, um documento recém-inserido de 100 KB gera no mínimo 200 KB de deltas em uma transação. (Essa é uma estimativa aproximada que não inclui metadados ou índices.)

Atualizações: qualquer atualização de documento, mesmo para um único campo, cria uma nova revisão de todo o documento no histórico, mais ou menos o delta da atualização. Isso significa

que uma pequena atualização em um documento grande ainda geraria um grande delta de transações. Por exemplo, adicionar 2 KB de dados em um documento existente de 100 KB cria uma nova revisão de 102 KB no histórico. Isso soma pelo menos 104 KB do total de deltas em uma transação. (Novamente, essa é uma estimativa aproximada que não inclui metadados ou índices.)

Exclusões: semelhante às atualizações, qualquer transação de exclusão cria uma nova revisão do documento no histórico. No entanto, a revisão `DELETE` recém-criada é menor do que o documento original porque tem dados de usuário nulos e contém apenas metadados.

Restrições de nomenclatura

A tabela a seguir descreve as restrições de nomenclatura na Amazon. QLDB

Nome do ledger	• Devem conter 1 a 32 caracteres alfanuméricos ou hifens. • Deve ter uma letra ou número para o primeiro e o último caracteres. • Não devem ser todos os números. • Não podem conter dois hifens consecutivos. • Diferencia maiúsculas de minúsculas.
Nome da transmissão do diário	
Nome da tabela	• As tags devem conter apenas 1-128 caracteres alfanuméricos ou sublinhados. • Deve ter uma letra ou um sublinhado para o primeiro caractere. • Pode conter qualquer combinação de caractere s alfanuméricos e sublinhados para os caracteres restantes. • Diferencia maiúsculas de minúsculas. • Não deve ser uma palavra reservada do QLDB PartiQL.

Informações QLDB relacionadas à Amazon

> **Important**
>
> Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com este serviço.

Tópicos

- Documentação técnica
- GitHub repositórios
- AWS postagens e artigos do blog
- Mídia
- AWS Recursos gerais

Documentação técnica

- Amazon QLDB FAQs — Perguntas frequentes sobre o produto.
- QLDBPreços da Amazon — informações e exemplos de AWS preços.
- AWS re:Post— fórum AWS comunitário para perguntas e respostas (Q&A).
- Amazon Ion: guias do desenvolvedor, guias de usuário e referências para o formato de dados Amazon Ion.
- PartiQL: um documento de especificação e tutoriais gerais para a linguagem de consulta partiQL.
- QLDBWorkshops — Workshops que fornecem exemplos práticos do uso da Amazon QLDB para criar system-of-record aplicativos, incluindo os seguintes laboratórios:
  - QLDBFundamentos da aprendizagem
  - Trabalhando com o Amazon Ion e convertendo Ion de e para JSON (Java)
  - Usando o AWS Glue Amazon Athena para habilitar QLDB dados para um data lake
  - Streaming de QLDB dados para uma instância de banco de dados Amazon Aurora My SQL

- Dados de qualidade à prova de adulteração usando a Amazon QLDB — uma implementação de AWS soluções que mostra como evitar que invasores adulterem dados de qualidade usando QLDB para manter um histórico preciso das alterações nos dados. AWS As implementações de soluções ajudam você a resolver problemas comuns e a criar mais rapidamente usando AWS.

GitHub repositórios

Drivers

- . NETmotorista — A. NETimplementação do QLDB driver.
- Go driver — Uma implementação Go do QLDB driver.
- Driver Java — Uma implementação Java do QLDB driver.
- Driver Node.js — Uma implementação Node.js do QLDB driver.
- Driver Python — Uma implementação do driver em Python. QLDB

Shell da linha de comando

- QLDBshell — Uma implementação em Python e Rust de uma interface de linha de comando para os QLDB dados transacionais. API

Aplicações de exemplo

- DMVAplicativo Java — Um aplicativo tutorial baseado em um caso de uso do departamento de veículos motorizados (DMV). Ele demonstra as operações básicas e as melhores práticas de uso QLDB e o QLDB driver do Java.
- . NETDMVaplicativo — Um aplicativo tutorial DMV baseado em que demonstra as operações básicas e as melhores práticas de uso QLDB e o QLDB driver para. NET.
- DMVAplicativo Node.js — Um aplicativo tutorial DMV baseado em que demonstra as operações básicas e as melhores práticas de uso QLDB e o QLDB driver do Node.js.
- Aplicativo Python — Um DMV aplicativo tutorial DMV baseado em Python que demonstra as operações básicas e as melhores práticas de uso QLDB e o driver do QLDB Python.
- Ledger loader — Uma estrutura Java para carregamento assíncrono de dados em um QLDB livro contábil em alta velocidade usando um canal de entrega compatível (Amazon,AWS DMS Amazon, SQS Kinesis Data StreamsSNS, Amazon ou). MSK EventBridge

- Processador de exportação — Uma estrutura Java extensível que gerencia o trabalho de processamento de QLDB exportações no Amazon S3 lendo a saída da exportação e iterando os blocos exportados em sequência.
- QLDBstreams sample Lambda em Python — Um aplicativo que demonstra como QLDB consumir streams usando uma função para enviar QLDB dados para AWS Lambda um SNS tópico da Amazon, que tem uma fila da Amazon inscrita nele. SQS
- QLDBamostra de OpenSearch integração de streams — Um aplicativo em Python que demonstra como integrar o OpenSearch Amazon Service QLDB usando streams.
- Aplicativo de entrada dupla — Um aplicativo Java que demonstra como modelar um aplicativo de contabilidade financeira de dupla entrada usando. QLDB
- QLDBKVSpara Node.js — Uma biblioteca simples de interface de armazenamento de valores-chave QLDB com funções extras para verificação de documentos.

Amazon Ion e PartiQL

- Bibliotecas Amazon Ion: a equipe do Ion apoiou bibliotecas, ferramentas e documentação.
- Implantação do partiQL: a implantação, a especificação e os tutoriais do partiQL.

AWS postagens e artigos do blog

- Como a Earnin criou seu serviço de contabilidade usando a Amazon QLDB (16 de fevereiro de 2023) — Descreve como a Earnin.com QLDB costumava criar um serviço contábil para fornecer aos usuários um aplicativo financeiro móvel moderno e completo.
- Exporte e analise dados de QLDB periódicos da Amazon usando o AWS Glue Amazon Athena (19 de dezembro de 2022) — Discute como você pode usar o recurso de exportação do with QLDB and AWS Glue Athena para fornecer recursos analíticos e de geração de relatórios às suas arquiteturas baseadas em livros contábeis.
- Como a Shinsegae International aprimora a experiência do cliente e evita a falsificação com a QLDB Amazon (3 de agosto de 2022) — Descreve como a Shinsegae International criou um serviço de verificação de autenticidade digital usando a QLDB Amazon para informar os clientes sobre a autenticidade de produtos de luxo e fornecer o histórico de compra e distribuição dos produtos.
- A BungKusit usa a ISV tecnologia da Amazon QLDB e da VeriDoc Global para melhorar a experiência do cliente e do agente de entrega (29 de abril de 2022) — Mostra como uma empresa

de logística, a BungKusit, costumava implementar uma plataforma centralizada e transparente QLDB para comunicação intersetorial com um registro de transações imutável e criptograficamente verificável.

- Use a Amazon QLDB como uma loja de valores-chave imutável com um REST API e JSON (14 de fevereiro de 2022) — Apresenta uma maneira de trabalhar QLDB como uma loja de valores-chave imutável e usar seus recursos de auditoria por meio de um. REST API
- Construindo um sistema bancário central com a Amazon QLDB (21 de janeiro de 2022) — Mostra como usar QLDB para criar um sistema de contabilidade bancária principal. Também mostra por que QLDB é um bom fit-for-purpose banco de dados que atende às necessidades do setor de serviços financeiros.
- Como a Specright usa QLDB a Amazon para criar uma rede de cadeia de suprimentos rastreável (21 de janeiro de 2022) — Mostra como a Specright usa QLDB para criar uma rede de cadeia de suprimentos rastreável que permite que marcas atacadistas, varejistas e fabricantes compartilhem dados críticos da cadeia de suprimentos e dados de especificações de embalagens.
- How f EMR entrega dados médicos criptograficamente seguros e verificáveis com a Amazon QLDB (23 de dezembro de 2021) — Explora como a Team f EMR usou QLDB e AWS gerenciou outros serviços para viabilizar seus esforços de socorro.
- Monitore os padrões de acesso às QLDB consultas da Amazon (8 de novembro de 2021) — Mostra como associar QLDB transações e as declarações partiQL que foram executadas nas transações às API solicitações recebidas pelo Amazon API Gateway.
- Crie uma CRUD operação simples e um fluxo de dados na Amazon QLDB usando AWS Lambda (28 de setembro de 2021) — Mostra como realizar CRUD (criar, ler, atualizar e excluir) operações QLDB usando AWS Lambda funções.
- Verificação criptográfica do mundo real com a Amazon QLDB (26 de agosto de 2021) — Discute o valor da verificação criptográfica em um QLDB livro contábil no contexto de um caso de uso realista.
- Verifique as condições de entrega com o Projeto Accord e a AmazonQLDB: Parte 1, Parte 2 (28 de junho de 2021) — Discute como aplicar a tecnologia de contratos legais inteligentes para verificar as condições de entrega com o Projeto Accord de código aberto e. QLDB
- Streaming de QLDB dados da Amazon via AWS CDK (7 de junho de 2021) — Mostra como usar o AWS Cloud Development Kit (AWS CDK) para configurarQLDB, preencher os QLDB dados usando AWS Lambda funções e configurar o QLDB streaming para fornecer resiliência de dados dos dados contábeis.

- Demonstre o controle de acesso refinado em QLDB (1º de junho de 2021) — Mostra como começar com permissões refinadas AWS Identity and Access Management (IAM) para um livro contábil. QLDB
- Processamento de streams com o DynamoDB Streams QLDB and Streams (23 de novembro de 2020) — fornece uma breve comparação entre streams e QLDB streams do DynamoDB e dicas sobre como começar a usá-los.
- Streaming de dados da Amazon QLDB para OpenSearch (19 de agosto de 2020) — Descreve como transmitir dados QLDB para o Amazon OpenSearch Service para oferecer suporte à pesquisa em rich text e à análise posterior, como agregação ou métricas entre registros.
- Como eu transmiti dados da Amazon QLDB para o DynamoDB usando o Nodejs quase em tempo real (7 de julho de 2020) — Descreve como transmitir dados para o DynamoDB para suportar latência de um dígito e consultas de QLDB valores-chave infinitamente escaláveis.
- Construindo uma interface GraphQL para a Amazon QLDB com AWS AppSync: Parte 1, Parte 2 (4 de maio de 2020) — Discute como integrar QLDB e AWS AppSync fornecer uma interface versátil, baseada em GraphQLAPI, em cima de um livro contábil. QLDB

Mídia

AWS vídeos

- AWS Tutoriais e demonstrações: para o QLDB Aurora Streaming (17 de março de 2023; 21 minutos) — Este vídeo explica os conceitos fundamentais para implementar a capacidade de streaming e demonstra como configurar o QLDB streaming de dados para um banco de dados downstream do QLDB Amazon Aurora My. SQL
- ArcBlock: Aproveitando QLDB a Amazon para criar uma solução de identidade descentralizada (31 de maio de 2022; 4 minutos) — ArcBlock explica a solução de identidade descentralizada que eles criaram usando. QLDB
- AWS re:Invent 2020: Usando a Amazon QLDB como system-of-trust banco de dados para os principais aplicativos de negócios (5 de fevereiro de 2021; 30 minutos) — Saiba como os primeiros QLDB usuários da Amazon aplicaram as propriedades exclusivas do banco de dados contábil para procedência de dados e verificabilidade criptográfica para implementar sistemas de registro com integridade de dados incorporada.
- AWS re:Invent 2020: Criando um aplicativo sem servidor com a QLDB Amazon (5 de fevereiro de 2021; 28 minutos) — Aprenda a criar, testar e otimizar um aplicativo sem servidor totalmente

funcional combinando a Amazon com serviços como AWS Lambda Amazon Kinesis e QLDB Amazon DynamoDB.

- AWS re:Invent 2020: Criação de aplicativos baseados em auditoria que mantêm a integridade dos dados com a Amazon QLDB (5 de fevereiro de 2021; 18 minutos) — Esta sessão aborda os problemas que a Amazon QLDB pode resolver, responde às suas perguntas sobre quando e por que você usaria um banco de dados contábil e compartilha casos de uso de clientes como a Osano.
- Como armazenar centralmente registros imutáveis usando a Amazon com. QLDB NETAplicativos (7 de dezembro de 2020; 10 minutos) — Uma demonstração de como usar QLDB com. NETaplicativos para armazenar centralmente registros imutáveis.
- Workshop virtual: Criação de sistemas de registro baseados em contabilidade com QLDB Java - Palestras técnicas AWS on-line (29 de julho de 2020; 87 minutos) — Um workshop conduzido por instrutor que percorre o laboratório do Working With Ion Immersion Day para criar um programa carregador de dados usando a biblioteca Amazon Ion e o driver para Java. QLDB
- Workshop para desenvolvedores: Usando o AWS Immutable Ledger Database QLDB no ABT Node (22 de junho de 2020; 34 minutos) — Um step-by-step guia sobre como instalar e configurar QLDB no Node. ABT Também explica como instalar e configurar o Blockchain Explorer e ArcBlock o Boarding Gate Blocklets aos quais se conectarQLDB.
- AWS Palestra técnica: a perspectiva de um cliente sobre a criação de um aplicativo de sistema de registro acionado por eventos com a QLDB Amazon (31 de março de 2020; 29 minutos) — Uma palestra técnica de Matt Lewis AWS , herói de dados e arquiteto-chefe da Agência de Licenciamento de Motoristas e Veículos DVLA () no Reino Unido — que DVLA explica o caso QLDB de uso e a arquitetura orientada a eventos de seu aplicativo.
- AWS re:Invent 2019: Por que você precisa de um banco de dados contábil:BMW,DVLA, & Sage discutem casos de uso (5 de dezembro de 2019; 47 minutos) — Uma apresentação de clientesBMW, da organização DVLA governamental do Reino Unido e da Sage que discute os motivos para usar um banco de dados contábil e compartilha seus casos de uso. QLDB
- AWS re:Invent 2019: AmazonQLDB: Uma análise aprofundada de um engenheiro sobre por que isso é um divisor de águas (5 de dezembro de 2019; 50 minutos) — Uma apresentação de Andrew Certain (engenheiro AWS ilustre) que discute a arquitetura exclusiva, que prioriza o diário, juntamente com suas várias inovações. QLDB Ele inclui hashing criptográfico, árvores Merkle, replicação de várias zonas de disponibilidade e suporte a partiQL.
- Criando aplicativos com a AmazonQLDB, um banco de dados contábil inédito - AWS Online Tech Talks (19 de novembro de 2019; 51 minutos) — Uma palestra técnica aprofundada que descreve os recursos exclusivos QLDB e os detalhes específicos sobre como usar a funcionalidade

principal. Isso inclui consultar o histórico completo de seus dados, verificar documentos criptograficamente e projetar um modelo de dados.

Podcasts

- Por que os clientes estão escolhendo a AmazonQLDB? (5 de julho de 2020; 33 minutos): uma discussão que explica a definição de um banco de dados ledger, como ele é diferente de um blockchain e como os clientes o estão usando atualmente.

AWS Recursos gerais

- Aulas e workshops — Links para cursos especializados e baseados em funções, além de laboratórios individualizados para ajudar a aprimorar suas AWS habilidades e ganhar experiência prática.
- AWS Centro do desenvolvedor — explore tutoriais, baixe ferramentas e saiba mais sobre eventos para AWS desenvolvedores.
- AWS Ferramentas para desenvolvedores — Links para ferramentas de desenvolvedorSDKs, IDE kits de ferramentas e ferramentas de linha de comando para desenvolver e gerenciar AWS aplicativos.
- Centro de recursos de introdução — Saiba como configurar seu aplicativo Conta da AWS, participar da AWS comunidade e lançar seu primeiro aplicativo.
- Tutoriais práticos — Siga os tutoriais para iniciar seu step-by-step primeiro aplicativo no. AWS
- AWS Whitepapers — Links para uma lista abrangente de AWS white papers técnicos, abrangendo tópicos como arquitetura, segurança e economia e criados por arquitetos de AWS soluções ou outros especialistas técnicos.
- AWS Support Center — O hub para criar e gerenciar seus AWS Support casos. Também inclui links para outros recursos úteis, como fóruns, informações técnicasFAQs, status de integridade do serviço AWS Trusted Advisor e.
- AWS Support— A principal página da web com informações sobre AWS Support um one-on-one canal de suporte de resposta rápida para ajudá-lo a criar e executar aplicativos na nuvem.
- Entrar em contato: um ponto central de contato para consultas relativas a faturas da AWS , contas, eventos, uso abusivo e outros problemas.
- AWS Termos do site — Informações detalhadas sobre nossos direitos autorais e nossa marca registrada; sua conta, licença e acesso ao site; e outros tópicos.

Histórico de lançamentos da Amazon QLDB

A tabela a seguir descreve mudanças importantes em cada versão da Amazon QLDB e as atualizações correspondentes no Amazon QLDB Developer Guide. Para receber notificações sobre atualizações desta documentação, você pode se inscrever no RSS feed.

- APIversão: 2019-01-02
- Última atualização da documentação: 3 de janeiro de 2023

Alteração	Descrição	Data
Depreciação do serviço	Aviso de fim do suporte: os clientes existentes poderão usar a Amazon QLDB até o final do suporte em 31/07/2025. Para obter mais detalhes, consulte Migrar um Amazon QLDB Ledger para o Amazon Aurora Postgre. SQL	18 de julho de 2024
IAMOrientação atualizada	Guia atualizado para se alinhar às IAM melhores práticas. Para obter mais informações, consulte Melhores práticas de segurança em IAM.	3 de janeiro de 2023
Atualização das políticas AWS gerenciadas	A Amazon QLDB atualizou as políticas AWS gerenciadas existentes `AmazonQLDBFullAccess` `AmazonQLDBConsoleFullAccess` e. Essas políticas têm uma nova permissão para permitir que as entidades principais editem revisões de documento	4 de novembro de 2022

	s usando um procedimento armazenado do PartiQL. Para obter mais informaçõ es, consulte políticas AWS gerenciadas para a Amazon QLDB.	
Edição de Dados	A Amazon QLDB agora oferece suporte ao procedime nto armazenado `REDACT_RE VISION` PartiQL para livros contábeis criados em ou após 22 de julho de 2021. Usando esse procedime nto armazenado, você pode excluir permanentemente as revisões inativas de documentos no histórico e ainda manter a integridade geral dos dados do seu ledger. Para obter mais informações, consulte Redação de revisões de documentos.	3 de novembro de 2022
Driver Node.js v3	O QLDB driver da Amazon para Node.js versão 3.0 agora está disponível ao público em geral. Esta versão apresenta suporte para a AWS SDK for JavaScript v3. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver -nodejsawslabs/.	26 de setembro de 2022

Driver Go v3	O QLDB driver da Amazon para Go versão 3.0 agora está disponível ao público em geral. Esta versão apresenta suporte para a AWS SDK for Go v2. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-goawslabs/.	11 de agosto de 2022
Novo editor de consultas em PartiQL	Um novo editor de consultas PartiQL no QLDB console da Amazon agora está disponível ao público em geral. O novo editor QLDB partiQL fornece uma interface aprimorada para criar consultas, depurar transações e explorar resultados. Para obter informações sobre como abrir e usar o editor, consulte Acessando a Amazon QLDB usando o console.	22 de junho de 2022

Mapeador de objetos de íons para. NETmotorista	O QLDB driver da Amazon para. NETa versão 1.3 introduz suporte para o mapeador de objetos Ion. Esse atributo permite que você ignore completam ente a necessidade de converter manualmente entre os tipos Amazon Ion e os tipos nativos de C#. Para ver o histórico completo de alterações do mapeador de objetos Ion, consulte o CHANGELOGarquivo.md no GitHub repositório. `amzn/ion-object-mapper-dotnet`	19 de janeiro de 2022
JSONformato de exportação de diário	A Amazon QLDB agora suporta o formato de saída JSONLines para exportaçõ es de periódicos. Para obter mais informações, consulte Exportação de dados de periódicos da Amazon QLDB.	21 de dezembro de 2021
Solução de problemas da Amazon QLDB	Foi adicionado um novo tópico de solução de problemas que fornece orientação para uma lista agregada de erros comuns que você pode encontrar ao usar a AmazonQLDB.	8 de dezembro de 2021

Lançamento de nova região	A Amazon agora QLDB está disponível na região do Canadá (Central). Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da Amazon Web Services	11 de novembro de 2021
Prevenção do problema de substituto confuso entre serviços	A Amazon QLDB agora suporta o uso das `aws:SourceArn` chaves de contexto de condição `aws:SourceAccount` global nas políticas de IAM recursos para evitar o confuso problema do deputado. Para obter mais informações, consulte Prevenção de substituto confuso entre serviços.	8 de novembro de 2021
Amazon QLDB shell v2	A versão 2.0 do QLDBshell da Amazon, escrita em Rust, agora está disponível ao público em geral. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-shellawslabs/.	14 de outubro de 2021

Atualização das políticas AWS gerenciadas	A Amazon QLDB atualizou as políticas AWS gerenciadas existentes `AmazonQLDBFullAccess` `AmazonQLDBConsoleFullAccess` e. Essas políticas têm permissões recém-adicionadas para permitir que os diretores passem qualquer recurso de IAM função em sua conta para o QLDB serviço. Isso é necessário para todas as solicitações de streams de diário. Para obter mais informações, consulte políticas AWS gerenciadas para a Amazon QLDB.	2 de setembro de 2021
AWS KMS Chaves gerenciadas pelo cliente	A Amazon QLDB agora oferece suporte à criptografia em repouso usando chaves gerenciadas pelo cliente em AWS Key Management Service (AWS KMS) para novos recursos contábeis. Para obter mais informações, consulte Criptografia em repouso na Amazon QLDB.	22 de julho de 2021

Atualização da política AWS gerenciada	A Amazon QLDB atualizou a política AWS gerenciad a existente `AmazonQLDBReadOnly` para remover uma `qldb:GetBlock` ação duplicada que foi listada anteriormente duas vezes. Para obter mais informaçõ es, consulte políticas AWS gerenciadas para a Amazon QLDB.	1º de julho de 2021
Lançamento de nova região	A Amazon agora QLDB está disponível na região Europa (Londres). Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da Amazon Web Services	24 de junho de 2021

Atualização das políticas AWS gerenciadas	A Amazon QLDB atualizou as políticas AWS gerenciad as existentes `AmazonQLDBFullAccess` `AmazonQLDBConsoleFullAccess` e. Essas políticas têm permissõe s recém-adicionadas para permitir que as entidades principais atualizem o modo de permissões em todos os ledgers e executem todos os comandos do PartiQL em todos os livros de permissõe s `STANDARD`. Para obter mais informações, consulte políticas AWS gerenciadas para a Amazon QLDB.	27 de maio de 2021
Permissões padrão disponíve is	A Amazon QLDB agora suporta um modo de `STANDARD` permissões para recursos contábeis. Com o modo de permissões padrão, você pode controlar o acesso com granularidade mais fina para ledgers, tabelas e comandos PartiQL. Para saber mais, consulte Conceitos básicos sobre o modo de permissões padrão.	27 de maio de 2021

Estatísticas de instruções PartiQL	O recurso de estatísticas de instruções partiQL agora está disponível no editor de consultas no console da AmazonQLDB. Para obter mais informações, consulte Obtendo estatísticas de instruções.	24 de maio de 2021
Tutorial: Verificando dados usando um AWS SDK	Foi adicionado um step-by-s tep tutorial com exemplos de código que demonstram como verificar um hash de revisão e um hash de bloco na Amazon QLDB usando o QLDB API por meio de um. AWS SDK Para saber mais, consulte Tutorial: Verificando dados usando um AWS SDK.	6 de maio de 2021
. NETdriver v1.2	O QLDB driver da Amazon para. NETA versão 1.2 agora está disponível ao público em geral. Esta versão introduz o APIs assíncrono. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-dotnetawslabs/.	1º de abril de 2021
Instruções `DROP INDEX` em PartiQL	O PartiQL na Amazon QLDB agora suporta a DROPINDEX declaração. Para obter mais informações, consulte Descartando índices.	3 de março de 2021

Estatísticas de instruções PartiQL	O recurso de estatísticas de instruções partiQL agora está disponível na versão mais recente do QLDB driver da Amazon para todos os idiomas suportados, inclusive . NET, Go e Python. Para obter mais informações, consulte Obtendo estatísticas de instruções.	25 de fevereiro de 2021
TypeScript tutorial de início rápido	Exemplos TypeScript de código adicionados no tutorial de início rápido do QLDB driver da Amazon para Node.js.	28 de dezembro de 2020
Estatísticas de instruções PartiQL	A versão mais recente do QLDB driver da Amazon para Java e Node.js agora fornece estatísticas de execução de instruções que podem ajudá-lo a executar instruçõe s partiQL com mais eficiência. Para obter mais informações, consulte Obtendo estatísticas de instruções.	22 de dezembro de 2020

Referências do cookbook do Driver e tutoriais de início rápido	Foram adicionadas referências de livros de receitas para os drivers Go e Node.js, tutoriais de início rápido para os drivers Java e Python e um guia para trabalhar com o Amazon Ion em. QLDB Para obter mais informações, consulte Introdução ao QLDB driver da Amazon.	24 de novembro de 2020
Amazon QLDB shell v1.1	A versão 1.1 do QLDBshell da Amazon agora está disponível ao público em geral. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-shellawslabs/.	26 de outubro de 2020
QLDBDriver da Amazon para Go	O QLDBdriver da Amazon para Go agora está disponível ao público em geral. Esse driver é de código aberto GitHub e permite que você use o AWS SDK for Go para interagir com os dados transacionais QLDB da. API Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-goawslabs/.	20 de outubro de 2020

Recomendações de configuração do driver Node.js	Foi adicionada uma nova seção que fornece recomendações de configuração para o QLDB driver da Amazon para Node.js, incluindo como reduzir a latência reutilizando conexões com o keep-alive.	16 de outubro de 2020
Criação de índice em tabelas não vazias	A Amazon QLDB agora oferece suporte à criação de índices em tabelas não vazias. Para obter mais informações, consulte Gerenciando índices.	30 de setembro de 2020
Otimizar a performance da consulta	Foi adicionada uma nova seção que descreve as restrições de consulta na Amazon QLDB e fornece orientação para otimizar o desempenho da consulta considerando essas restrições.	18 de setembro de 2020
Referência do Cookbook para o driver Java	Foi adicionada uma referência ao Cookbook que mostra exemplos de código de casos de uso comuns do QLDB driver Amazon para Java.	3 de setembro de 2020
Gerenciamento da sessão com o driver	Foi adicionada uma nova seção que fornece uma visão geral do gerenciamento de sessões com o motorista na Amazon QLDB e descreve como o QLDB motorista lida com as sessões ao executar transações de dados.	1.º de setembro de 2020

Driver Java v2.0	O QLDB driver da Amazon para Java versão 2.0 agora está disponível ao público em geral. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-javaawslabs/.	28 de agosto de 2020
Driver Node.js v2.0	O QLDB driver da Amazon para Node.js versão 2.0 agora está disponível ao público em geral. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-nodejsawslabs/.	27 de agosto de 2020
Informações relacionadas	Foi adicionado um novo tópico que contém links para informações relacionadas e recursos adicionais para ajudar você a entender e trabalhar com a AmazonQLDB.	24 de agosto de 2020
Driver em Python v3.0	O QLDB driver Amazon para Python versão 3.0 agora está disponível ao público em geral. Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-pythonawslabs/.	20 de agosto de 2020

QLDBDriver da Amazon para Go	Uma versão prévia do QLDB driver da Amazon para Go já está disponível. Esse driver permite que você use o AWS SDK for Go para interagir com os dados transacionais QLDB da. API Para obter mais informações, consulte QLDBDriver da Amazon para Go (versão prévia).	6 de agosto de 2020
Referência do Cookbook para o driver Python	Foi adicionada uma referênci a ao Cookbook que mostra exemplos de código de casos de uso comuns do QLDB driver Amazon para Python.	24 de julho de 2020
Único IDs na Amazon QLDB	Foi adicionada uma nova seção que descreve as propriedades e as diretrizes de uso do Unique IDs na Amazon QLDB.	9 de julho de 2020
AWS CloudFormation recurso para fluxos de periódicos	A Amazon QLDB agora oferece suporte a um recurso para criar fluxos de diários usando um AWS CloudForm ation modelo. Para obter mais informações, consulte o recurso AWS::QLDB: :Stream no Guia do AWS CloudForm ation usuário.	9 de julho de 2020

Referência do livro de receitas para o. NETmotorista	Foi adicionada uma referência ao livro de receitas que mostra exemplos de código de casos de uso comuns do QLDB driver da Amazon para. NET.	1º de julho de 2020
QLDBDriver da Amazon para. NET	O QLDBdriver da Amazon para. NETagora está disponível ao público em geral. Esse driver é de código aberto GitHub e permite que você use o AWS SDK for .NET para interagir com os dados transacionais QLDB da. API Para notas de lançamento, consulte o GitHub repositório amazon-qldb-driver-dotnetawslabs/.	26 de junho de 2020
QLDBDriver da Amazon para Node.js	O QLDBdriver da Amazon para Node.js agora está disponível ao público em geral. Esse driver é de código aberto GitHub e permite que você use o AWS SDK for JavaScript no Node.js para interagir com os dados transacionais QLDB do. API Para notas de lançamento, consulte o repositório awslabs/amazon-qldb-driver-nodejs GitHub .	5 de junho de 2020

Streams de QLDB periódicos da Amazon	A Amazon QLDB agora permite que você crie um stream que captura todas as revisões de documento s comprometidas com seu diário e entrega esses dados ao Amazon Kinesis Data Streams quase em tempo real. Para obter mais informações, consulte Dados de periódico s de streaming da Amazon QLDB.	19 de maio de 2020
Exemplos de código do Amazon Ion	Foram adicionados exemplos de código que consultam e processam dados do Amazon Ion em um QLDB livro contábil da Amazon usando o QLDB driver para Java, Node.js e Python. Para obter mais informações, consulte exemplos de código Ion na Amazon QLDB.	12 de maio de 2020
Modelo de simultaneidade e conteúdo do diário	Expandiu o tópico do modelo de QLDB simultaneidade da Amazon para adicionar informações sobre o uso de índices para limitar conflitos otimistas de controle de simultaneidade (). OCC Foi adicionada uma nova seção que descreve o conteúdo do periódico na Amazon QLDB.	4 de maio de 2020

Guia de início rápido para o driver Node.js	Foi adicionado um guia de início rápido para o QLDB driver da Amazon para Node.js.	1º de maio de 2020
QLDBConcha da Amazon	O Amazon QLDB Shell agora está disponível ao público em geral. Esse shell é de código aberto GitHub e fornece uma interface de linha de comando que permite executar instruções partiQL em dados contábeis . Para notas de lançamento, consulte o repositório awslabs/amazon-qldb-shell GitHub .	20 de abril de 2020
Certificações de compatibilidade	A Amazon agora QLDB é certificada para programas de AWS conformidade, incluindo HIPAA ISO e. Para obter mais informações, consulte Validação de conformidade para a Amazon QLDB.	3 de abril de 2020
Recomendações expandidas para driver	Expandiu a seção de recomendações de QLDB drivers da Amazon para aplicá-la a todas as linguagens de programação suportadas.	2 de abril de 2020

QLDBConcha da Amazon	Uma versão prévia do Amazon QLDB Shell já está disponíve l. Esse shell fornece uma interface de linha de comando que permite executar instruçõe s partiQL em dados de ledger. Para obter mais informações, consulte Acessando a Amazon QLDB usando o QLDB shell (APIsomente dados) (versão prévia).	23 de março de 2020
Driver Java v1.1	O QLDB driver da Amazon para Java versão 1.1 agora está disponível ao público em geral. Para notas de lançamento, consulte o repositório awslabs/ amazon-qldb-driver-java GitHub .	20 de março de 2020
QLDBDriver da Amazon para. NET	A Amazon QLDB agora fornece uma versão prévia do. NETmotorista. Esse driver permite que você use o AWS SDK for .NET para interagir com os dados transacionais QLDB da. API Para obter mais informações, consulte Amazon QLDB driver for. NET(pré-v isualização).	13 de março de 2020

QLDBDriver Amazon para Python	O QLDBdriver da Amazon para Python agora está disponível ao público em geral. Esse driver é de código aberto GitHub e permite que você use o AWS SDK for Python (Boto3) para interagir com os dados transacionais QLDB da. API Para notas de lançamento, consulte o repositório awslabs/ amazon-qldb-driver-python GitHub .	11 de março de 2020
Instrução partiQL e `UNDROP TABLE` aninhada DML	O PartiQL na Amazon QLDB agora suporta instruções de linguagem de manipulação de dados (DML) nas quais coleções aninhadas são especificadas como fontes na cláusula. `FROM` Para obter mais informações, consulte a FROMdeclaração na referência do partiQL. QLDBO partiQL também suporta a UNDROPTABLEdeclaração. Para obter mais informações, consulte Cancelando a remoção de tabelas.	26 de fevereiro de 2020
Tópico de introdução expandido	Expandiu a introdução O que é a Amazon QLDB? tópico para incluir as novas seções Visão geral da Amazon QLDB e Do relacional ao livro contábil.	24 de janeiro de 2020

Referência partiQL para funções suportadas	Expandiu a referência do Amazon QLDB PartiQL para incluir informações detalhada s de uso sobre as funções suportadasSQL. Para obter mais informações, consulte funções PartiQL.	2 de janeiro de 2020
Recomendações de drivers e erros comuns	Foram adicionadas novas seções que descrevem recomendações de drivers para o QLDB driver Amazon para Java e erros comuns para todas as linguagens de driver.	2 de janeiro de 2020
Lançamento de novas regiões	A Amazon agora QLDB está disponível nas regiões Ásia-Pacífico (Seul), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney) e Europa (Frankfurt). Para obter uma lista completa das regiões disponíveis, consulte QLDBendpoints e cotas da Amazon no. Referência geral da Amazon Web Services	19 de novembro de 2019

QLDBDriver da Amazon para Node.js	A Amazon QLDB agora fornece uma versão prévia do driver Node.js. Esse driver permite que você use o AWS SDK for JavaScript no Node.js para interagir com os dados transacionais QLDB do. API Para obter mais informaçõ es, consulte o QLDBdriver da Amazon para Node.js (versão prévia).	13 de novembro de 2019
QLDBDriver Amazon para Python	A Amazon QLDB agora fornece uma versão prévia do driver Python. Esse driver permite que você use o AWS SDK for Python (Boto3) para interagir com os dados transacionais QLDB da. API Para obter mais informaçõ es, consulte QLDBDriver da Amazon para Python (versão prévia).	29 de outubro de 2019
Versão pública	Este é o lançamento público inicial da AmazonQLDB. Esta versão inclui o Guia do Desenvolvedor e a APIreferê ncia integrada de gerenciam ento de livros contábeis.	10 de setembro de 2019

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.