aws

开发人员指南

Amazon Quantum Ledger Database (Amazon QLDB)

Amazon Quantum Ledger Database (Amazon QLDB): 开发人员指南

Table of Contents

什么是 Amazon QLDB？

Amazon Quantum Ledger Database（Amazon QLDB）是一个完全托管的分类账数据库，提供了一个透明、不可变、可通过加密方式验证的事务日志，且该事务日志由一家可信的中央机构拥有。您可以使用 Amazon QLDB 来跟踪所有应用程序数据更改，并不断维护完整且可验证的更改历史记录。要详细了解 Amazon Web Services 上提供的各种数据库选项，请参阅为 AWS 上的组织选择合适的数据库。

分类账通常用于记录组织中的经济和金融活动历史记录。许多组织构建具有类似分类账功能的应用程序，因为他们希望维护应用程序数据的准确历史记录。例如，他们可能想要跟踪银行事务中的贷方和借方历史记录，验证保险索赔的数据谱系，或跟踪供应链网络中项目的移动。分类帐应用程序通常通过在关系数据库中创建的自定义审计表或审计跟踪记录来实现。

Amazon QLDB 是一种全新的数据库，您无需参与构建类似分类账的应用程序的复杂开发工作。使用 QLDB，您的数据更改历史是无法更改的，无法在原地覆盖或更改。而且，您可通过加密技术，以验证应用程序的数据是否没有发生意外更改。QLDB 使用不可改变的事务日志，即日记账。该日记账仅限追加，由一组有序和哈希链的数据块组成，其中包含您提交的数据。

Amazon QLDB 视频

要了解 Amazon QLDB 及其如何使您受益，请在 YouTube 上观看这段QLDB 概述视频。

Amazon QLDB 定价

使用 Amazon QLDB，您只需按使用量付费，没有最低费用或强制使用服务。您只需为您的分类账数据库使用的资源支付费用即可，无需提前进行预配置。

有关更多信息，请参阅 Amazon QLDB 定价。

QLDB 入门

我们建议您首先阅读以下主题：

- Amazon QLDB 概览 — 了解 QLDB 的高级概述。
- Amazon QLDB 中的核心概念和术语 — 学习 QLDB 的基本概念和术语。
- 访问 Amazon QLDB — 了解如何使用AWS Management Console、API 或AWS Command Line Interface (AWS CLI) 访问 QLDB。

- Amazon MQ 如何与 IAM 协同工作 — 学习如何AWS Identity and Access Management使用 (IAM) 控制对 QLDB 的访问权限。

要快速开始使用 QLDB 控制台，请参阅Amazon QLDB 控制台入门。

要了解如何使用AWS提供的驱动程序使用 QLDB 进行开发，请参阅Amazon QLDB 驱动程序入门。

Amazon QLDB 概览

以下章节高度概述了 Amazon QLDB 服务组件及其交互方式。

主题

- 日记账优先
- 不可变的
- 可通过密码验证
- 类似 SQL、且文档灵活
- 开源开发人员工具
- 无服务器且高度可用
- 企业级

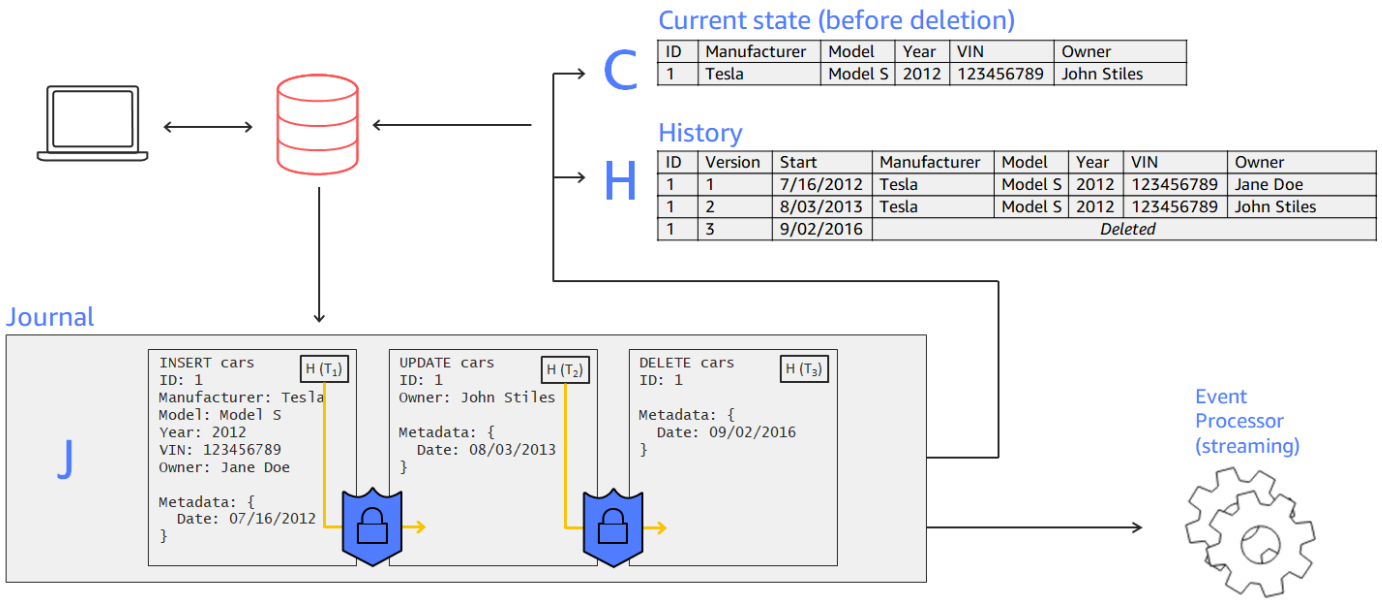
日记账优先

在传统的数据库架构中，通常将数据写入表中作为事务的一部分。事务日志 —（通常是内部实现）记录了所有事务及其对数据库所做的修改。事务日志是数据库的重要组成部分。在发生系统故障、灾难恢复或者数据复制时，您需要使用日志来重播事务。但是，数据库事务日志并非一成不变，也不是为用户提供直接、便捷的访问而设计的。

在 Amazon QLDB 中，日记账是数据库的核心。此日记账在结构上与事务日志类似，是一种不可变的、仅限附加的数据结构，用于存储您的应用程序数据以及相关的元数据。所有写入事务（包含更新和删除）都将首先提交到日记账中。

QLDB 使用日记账确定分类账数据的当前状态，方法是将其具体化为可查询的、用户定义的表格。这些表格还提供了所有事务数据（包含文档修订版和元数据）的可访问历史记录。此外，该日记账还处理分类账数据的并发性、排序、加密验证和可用性。

下图所示为 QLDB 日记账体系结构。



- 在此示例中，应用程序连接到分类账并运行在名为`cars`的表中插入、更新和删除文档的事务。
- 数据首先按顺序写入日记账。
- 然后，使用内置视图将数据具体化至表中。您可通过这些视图查询汽车的当前状态和完整历史记录，并为每个版本分配一个版本号。
- 您也可直接从日记中导出或流式传输数据。

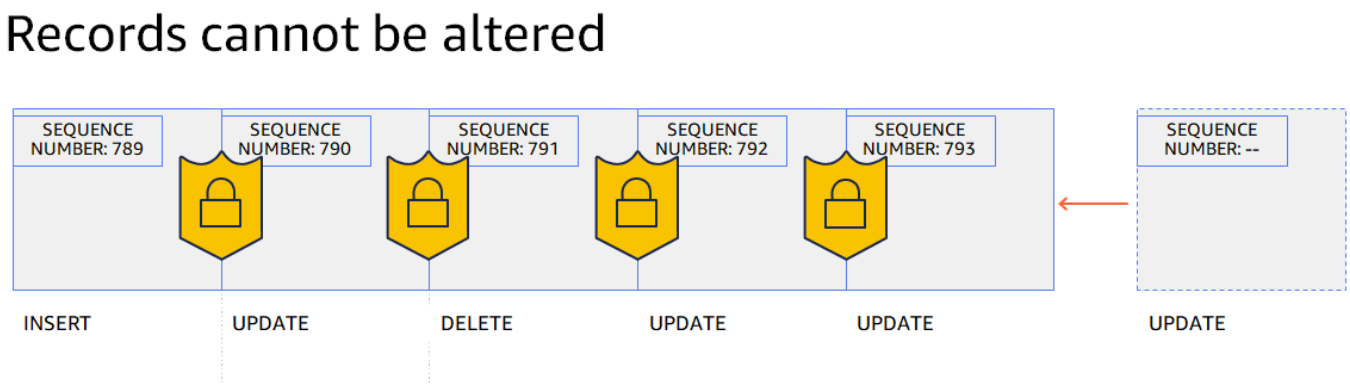
不可变的

由于 QLDB 日记账仅限追加，因此它会完整记录所有无法修改或者覆盖的数据更改。没有 API 或其他方法可更改任何已提交的数据。这种日记账结构允许您访问和查询分类账完整历史记录。

> **Note**
>
> QLDB 支持不可变性的唯一例外是数据编修。使用此功能，您可以遵守监管法规，例如欧盟的《通用数据保护条例》(GDPR) 与《加州消费者隐私法》(CCPA)。
>
> QLDB 还支持数据编修操作，允许您永久删除表历史记录中的非活动文档的修订内容。密文操作仅删除指定修订版中的用户数据，而日记账序列和文档元数据则保持不变。这样可保持分类账的整体数据完整性。有关更多信息，请参阅对文档修订版执行编校。

QLDB 在事务中向日记账中写入一个数据块。每个块都包含代表您插入、更新和删除的文档修订版本的条目对象，以及提交这些修订的语句。这些数据块是按顺序排列以及哈希链的，以保证数据的完整性。

下面的示意图阐明了此日记账结构。

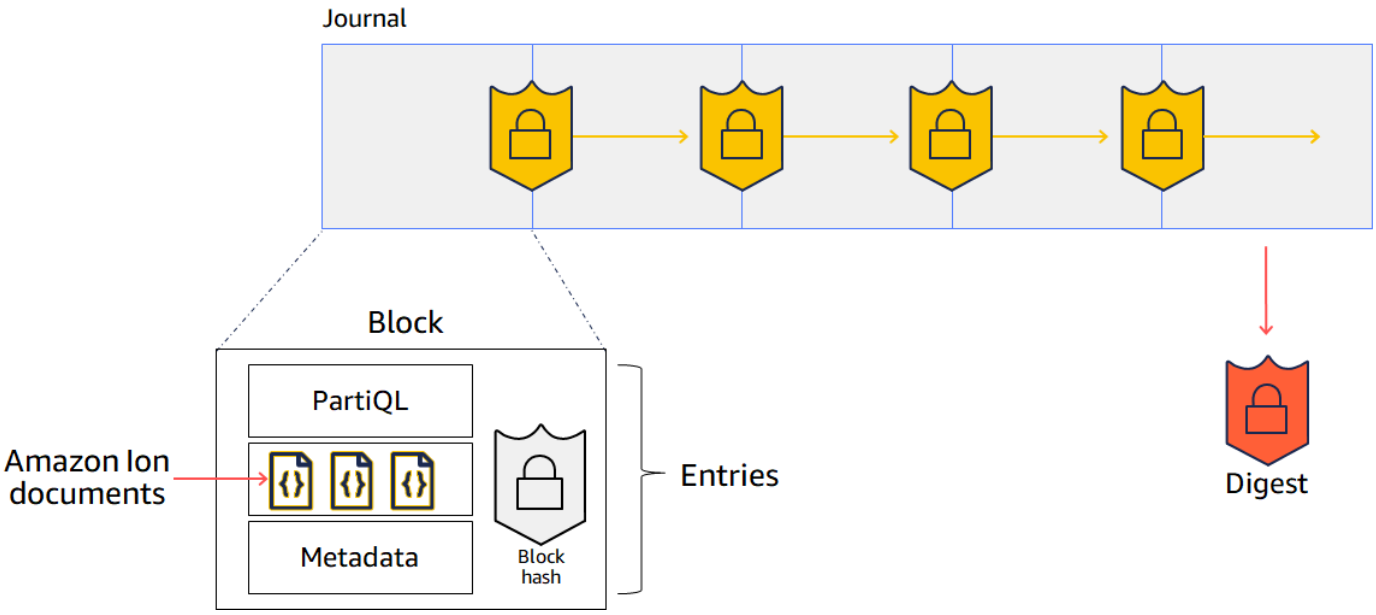


该图显示，事务以数据块的形式提交到日记账中，这些数据块被哈希链连接以供验证。每个数据块都有一个序列号来指定其地址。

可通过密码验证

日记账数据块通过加密哈希技术进行排序并链接在一起，类似于数据块链。QLDB 使用日记账哈希链，通过加密验证方法提供事务数据的完整性。使用摘要（代表日记账截至某个时间点的完整哈希链的哈希值）和默克尔审计证明（一种证明二叉哈希树中任何节点有效性的机制），您可以验证您的数据在任何时候都没有发生意想不到的变化。

下图显示了涵盖日记账在某个时间点的完整哈希链的摘要。



在此图中，使用 SHA-256 加密哈希函数对日记账数据块进行哈希处理，并按顺序链接到后续块。每个数据块都包含包含您的数据文档、元数据和事务中运行的 PartiQL 语句的条目。

有关更多信息，请参阅Amazon QLDB 中的数据验证。

类似 SQL、且文档灵活

QLDB 使用 PartiQL 作为其查询语言，使用 Amazon Ion 作为其面向文档的数据模型。PartiQL 是开源、与 SQL 兼容的查询语言，现已扩展为可与 Ion 配合使用。使用 PartiQL，您可使用熟悉的 SQL 运算符插入、查询和管理数据。查询平面文档时，语法与使用 SQL 查询的关系表相同。要了解有关 PartiQL 的 QLDB 实现的更多信息，请参阅Amazon QLDB PartiQL 参考。

Amazon Ion 是 JSON 的超集。Ion 是基于文档的开源数据格式，可让您灵活地存储和处理结构化、半结构化和嵌套数据。要了解有关 QLDB 中 Ion 的更多信息，请参阅 Amazon QLDB 中的 Amazon Ion 数据格式参考。

有关传统关系数据库与 QLDB 中核心组件和功能的高级比较，请参阅从关系至分类账。

开源开发人员工具

为了简化应用程序开发，QLDB 提供了各类编程语言的开源驱动程序。您可使用这些驱动程序通过在分类账上运行 PartiQL 语句，并处理这些语句的结果来与事务数据 API 进行交互。有关当前支持的驱动程序语言的信息和教程，请参阅 Amazon QLDB 驱动程序入门。

Amazon Ion 还提供可为您处理 Ion 数据客户端库。有关处理 Ion 数据的开发者指南和代码示例，请参阅 GitHub 上的 Amazon Ion 文档。

无服务器且高度可用

QLDB 完全托管、无服务器、且高度可用。该服务会自动扩展，以支持您的应用程序需求，您无需预配置实例或容量。您的数据的多个副本将在AWS 区域中的可用区内复制，并在中的可用区之间复制。

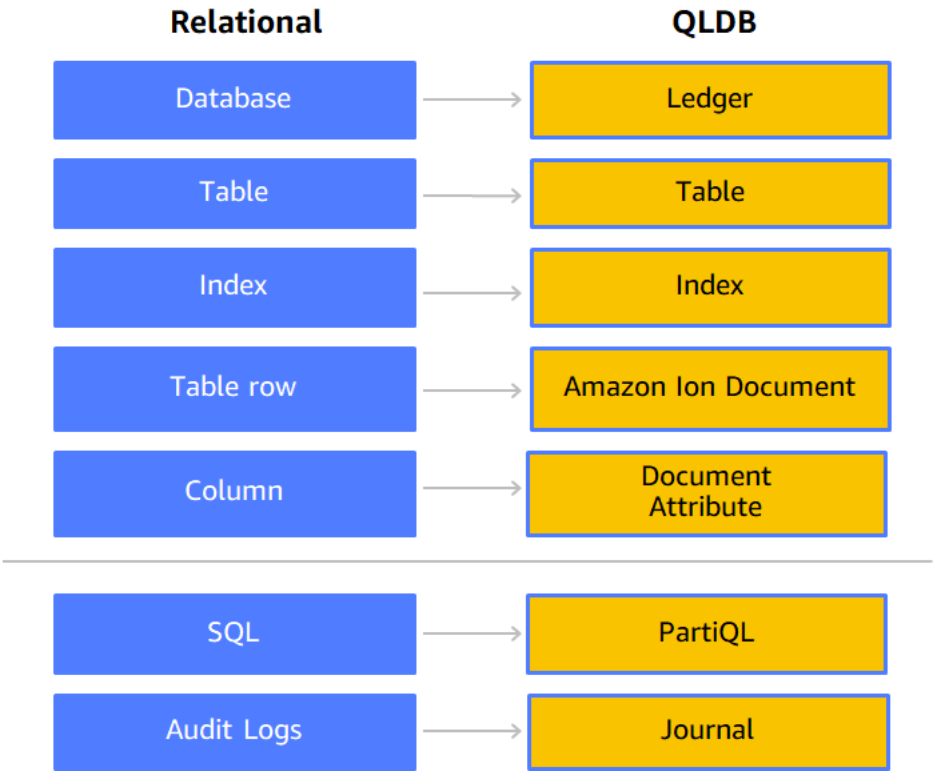
企业级

QLDB 事务完全符合原子性、一致性、隔离性与持久性 (ACID) 属性。QLDB 使用乐观并发控制 (OCC)，事务以完全序列化性运行，即最高级别的隔离。这意味着不存在出现幻像读取、脏读、写入偏差或者其他类似并发问题的风险。有关更多信息，请参阅Amazon QLDB 并发模型。

从关系至分类账

如果您是应用程序开发人员，则可能在使用关系数据库管理系统 (RDBMS) 和结构化查询语言 (SQL) 方面有一些经验。在您开始使用 Amazon QLDB，您既会遇到许多相似之处，也会遇到许多相似之处。随着您进入更高级的话题，您还将遇到 QLDB 在 RDBMS 基础上构建的强大新功能。本节介绍常见数据库组件和操作，并与其 QLDB 中的等效操作进行比较和对比。

下图介绍传统 RDBMS 和 Amazon QLDB 之间核心组件的映射结构。



下表介绍了传统 RDBMS 和 QLDB 之间内置操作功能的主要高级相似之处和不同之处。

操作	RDBMS	QLDB
创建表	定义所有列名和数据类型的CREATE TABLE 语句	未定义任何表属性或数据类型以允许无架构和开放内容的CREATE TABLE 语句
创建索引	CREATE INDEX statement	表中任何顶级字段的CREATE INDEX 语句
插入数据	INSERT 语句，它指定新行或元组中的值，该行或元组符合表所定义的架构	INSERT 语句，以任何有效的 Amazon Ion 格式指定新文档中的值，无论表格中是否存在现有文档
查询数据	SELECT-FROM-WHERE statement	查询平面文档时语法与 SQL 相同语法的SELECT-FROM-WHERE语句

操作	RDBMS	QLDB
更新数据	UPDATE-SET-WHERE statement	UPDATE-SET-WHERE 更新平面文档时语法与 SQL 相同语法的语句
删除数据	DELETE-FROM-WHERE statement	DELETE-FROM-WHERE 删除平面文档时语法与 SQL 相同语法的语句
嵌套与半结构化数据	仅限扁平行或者元组	可以纳入 Amazon Ion 数据格式和 PartiQL 查询语言支持的任何结构化、半结构化或嵌套数据的文档
查询元数据	无内置元数据	从表的内置已提交视图中查询的SELECT语句
查询修订历史记录	无内置数据历史记录	从内置历史函数进行查询的SELECT语句
加密验证	没有内置密码学或不可变性	返回日记摘要，以及验证任何文档修订版本相对于该摘要完整性的证明的 API

有关 QLDB 中的核心概念和术语的概述，请参阅 核心概念。

有关在分类账中创建、查询和管理数据的过程的详细信息，请参阅 处理数据与历史记录。

Amazon QLDB 中的核心概念和术语

本节概述了 Amazon QLDB 中的核心概念和术语，包括分类账结构和分类账如何管理数据。作为分类账数据库，QLDB 与其他面向文档的数据库的不同之处在于以下关键概念。

主题

- QLDB 数据对象模型
- 日记账账优先事务
- 查询数据

- 数据存储
- QLDB API 模型
- 后续步骤

QLDB 数据对象模型

Amazon QLDB 中的基础数据对象模型说明如下：

1. 分类账

   您的第一步是创建分类账，这是 QLDB 中的主要 AWS 资源类型。要了解如何创建分类账，请参阅控制台入门中的第 1 步：创建新分类账或Amazon QLDB 分类账的基本操作。

   对于分类账的 ALLOW_ALL 和 STANDARD 权限模式，您可以创建 AWS Identity and Access Management (IAM) 策略，以授予在此分类账资源上运行 API 操作的权限。

   分类账 ARN 格式：

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

2. 日记账账和表格

   要开始在 QLDB 分类账中写入数据，首先要创建一个包含基本CREATE TABLE语句的表格。分类账数据由提交至分类账日记账文档的修订版组成。您可以在用户定义表格的上下文中将文档修订提交至分类账。在 QLDB 中，表格表示日记账中文档修订集合的实体化视图。

   在分类账的 STANDARD 权限模式下，您必须创建 IAM policy 来授予在此表资源上运行 PartiQL 语句的权限。凭借对表资源的权限，您可以运行访问表当前状态的语句。您也可以使用内置 `history()` 函数查询表的修订历史记录。

   表格 ARN 格式：

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

   有关向分类账及其关联资源授予权限的更多信息，请参阅 Amazon MQ 如何与 IAM 协同工作。

3. 文档

   表由的修订版QLDB 文档组成，这些版本是 Amazon Ion `struct` 格式的数据集。文档修订版表示由唯一文档 ID 标识的一系列文档的单一版本。

QLDB 存储您所提交文档的完整更改历史记录。表格允许您查询其文档的当前状态，而该 `history()` 函数允许您查询表格文档的整个修订历史记录。有关查询和编写修订版的详细信息，请参阅 处理数据与历史记录。

4. 系统目录

   每个分类账还提供系统定义的目录资源，您可通过查询该资源列出分类账中的所有表和索引。在分类账的 STANDARD 权限模式下，您需要拥有该目录资源的 `qldb:PartiQLSelect` 权限才能执行以下操作：

   - 在系统目录表 information_schema.user_tables上运行SELECT语句。
   - 在 QLDB 控制台的分类账详情页面上查看表格和索引信息。
   - 在 QLDB 控制台的 PartiQL 编辑器中查看表和索引列表。

   目录 ARN 格式：

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

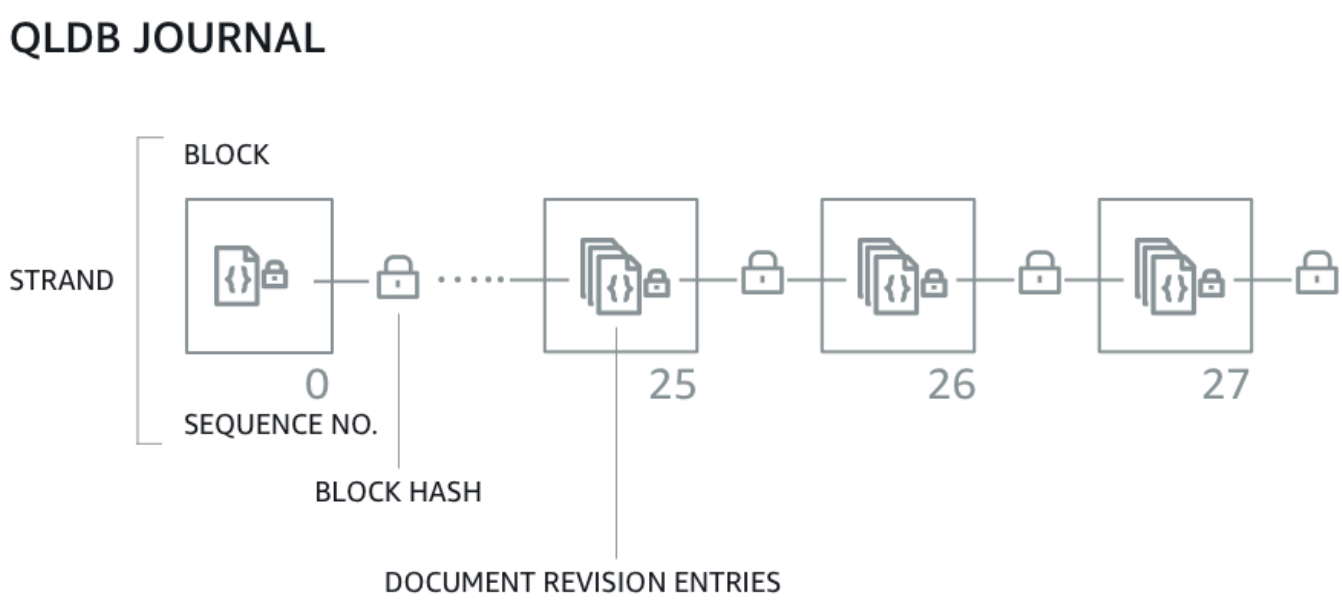
日记账账优先事务

当应用程序在 QLDB 分类账中读取或写入数据时，它是在数据库事务中读取或写入数据的。所有事务均受Amazon QLDB 资源中的限额和限制中定义的限制。在事务中，QLDB 执行以下步骤：

1. 分类账读取数据的当前状态。
2. 执行事务中提供的语句，然后使用乐观并发控制 (OCC) 检查其是否存在任何冲突，以确保完全可序列化的隔离。
3. 如果未发现 OCC 冲突，则按如下方式返回事务结果：
   - 对于读取，返回结果集并以仅附加 SELECT 的方式将语句提交至日记账中。
   - 对于写入，请以仅限追加的方式将任何更新、删除或新插入的数据提交到日记账中。

该日记账代表了关于所有日记账更改的、完整且不可变的历史记录。QLDB 在事务中向日记账中写入一个链式数据块。每个块都包含代表您插入、更新和删除的文档修订版本的条目对象，以及提交这些修订的 PartiQL 语句。

下面的示意图阐明了此日记账结构。



该图显示事务作为包含文档修订条目的数据块提交到日记账账中。每个数据块都经过哈希处理，并链接到后续数据块进行验证。每个数据块都有一个序列号，用于指定其在链中的地址。

> **Note**
>
> 在 Amazon QLDB 中，分支是分类账日记账账的分区。QLDB 目前仅支持单链日记账。

有关数据块的内容的更多信息，请参阅 Amazon QLDB 中的日记账内容。

查询数据

QLDB 旨在满足高性能联机事务处理 (OLTP) 的工作负载需求。分类账根据提交至日记账账的事务信息，提供数据的可查询表格视图。QLDB 中的表格视图是表中数据的子集。视图采用实时维护，因此它们始终可供应用程序查询。

您可以使用 PartiQL SELECT 语句查询以下系统定义的视图：

- 用户 - 仅包含您在表中写入的数据的最新有效版本（即用户数据的当前状态）。这是 QLDB 中的默认视图。
- 已提交 – T您的用户数据和系统生成的元数据的最新有效版本。这是与您用户表直接对应的完整系统定义表。

除了这些可查询视图外，您还可以使用内置历史记录函数查询数据的修订历史记录。历史函数在与已提交视图相同的架构中返回您的用户数据和关联的元数据。

数据存储

QLDB 包含两种类型的数据存储：

- 日记账存储 - 分类账日记账账使用的磁盘空间。该日记账仅限附录，包含所有数据更改的完整、不可变且可验证历史记录。
- 索引存储 - 分类账表、索引和索引历史记录使用的磁盘空间。索引存储包含的分类账数据针对高性能查询进行了优化。

将数据提交至日记账账后，它会具体化至您定义的表中。这些表经过优化，可实现更快、更高效查询。当应用程序使用事务数据 API 读取数据时，它会访问存储在索引存储中的表格和索引。

QLDB API 模型

QLDB 提供了两类 API，您的应用程序代码可以与之交互：

- Amazon QLDB — QLDB 资源管理 API（也称为控制面板）。此 API 仅用于管理分类账资源和非事务数据操作。您可以使用这些操作创建、删除、描述、列出和更新分类账。您还可以通过加密方式验证日记账数据，并导出或流式传输日记账块。
- Amazon QLDB 会话 — QLDB 事务数据 API。您可以使用此 API 在带有 PartiQL 语句的分类账上运行数据事务。

> **Important**
>
> 我们建议使用 QLDB 驱动程序或 QLDB Shell 在分类账上运行数据交易，而不是直接与 QLDB 会话 API 交互。
>
> - 如果您在使用 AWS SDK，请使用 QLDB 驱动程序。该驱动程序在 QLDB 会话数据 API 之上提供了一个高级抽象层，并为您管理SendCommand操作。有关信息和支持的编程语言列表，请参阅 驱动程序入门。
> - 如果您正在使用 AWS CLI，请使用 QLDB Shell。Shell 是一个命令行接口，它使用 QLDB 驱动程序与分类账进行交互。有关信息，请参阅 使用 Amazon QLDB Shell (仅限数据 API)。

有关这些 API 操作的更多信息，请参阅 Amazon QLDB API 参考。

后续步骤

要了解如何使用分类账处理数据，请参阅 在 Amazon QLDB 中处理数据与历史记录 并遵循描述创建表、插入数据和运行基本查询过程的示例。本指南使用示例数据和查询示例作为上下文，详细说明这些概念的运行方式。

要使用 QLDB 控制台快速开始使用示例应用程序教程，请参阅Amazon QLDB 控制台入门。

有关本节中描述的关键术语和定义的列表，请参阅 Amazon QLDB 词汇表。

Amazon QLDB 中的日记账内容

在 Amazon QLDB 中，日记账是不可变的事务记录，用于存储数据的所有完整且可验证的更改历史记录。该日记账仅限追加，由一组有序和哈希链的数据块组成，其中包含您提交的数据和其他系统元数据。QLDB 在事务中向日记账中写入一个链式数据块。

本节介绍了包含样本数据的日记账数据块的示例，并描述了数据块的内容。

主题

数据块示例

日记账数据块包含事务元数据、以及代表事务中提交的文档修订版本条目、和提交这些修订的 PartiQL 语句。

以下是包含示例数据的数据块示例。

> **Note**
>
> 数据块示例仅用于参考。显示的哈希值并非实际计算的哈希值。

```
{
```

```
  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"
      }
    }
  ]
}
```

在revisions字段中，某些修订版本对象可能只包含一个 hash 值而不包含其他属性。这是仅供内部使用的系统修订版，不包含用户数据。修订版的哈希值是该日记账完整哈希链的一部分，这是加密验证的必要条件。

数据块内容

日记账数据块包含以下字段：

blockAddress

日记账中的数据库位置。地址是一种包含两个字段的 Amazon Ion 结构：即strandId和sequenceNo。

例如：{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}

transactionId

提交数据块的事务唯一 ID。

blockTimestamp

数据块提交到日记账的时间戳。

blockHash

唯一代表数据块的 256 位哈希值。这是entriesHash和 previousBlockHash的串联的哈希值。

entriesHash

代表数据块内所有条目的哈希值，包括仅限内部的系统条目。这是 Merkle tree的根哈希，其中叶节点由entriesHashList中的所有哈希组成。

previousBlockHash

日记账中前一个链式数据块的哈希值。

entriesHashList

代表数据块中每个条目的哈希列表。此列表可以包含以下条目哈希：

- 表示transactionInfo的 Ion 哈希。该值是通过取整个 transactionInfo 结构的 Ion 哈希值计算得出的。
- Merkle tree 的根哈希，其中叶节点由revisions中的所有哈希组成。
- 表示redactionInfo的 Ion 哈希。此哈希值仅存在于由编校事务提交的数据块中。它的值是通过取整个 redactionInfo 结构的 Ion 哈希值计算得出的。
- 代表仅限内部使用的系统元数据哈希。这些哈希值可能并不存在于所有数据块。

transactionInfo

一种 Amazon Ion 结构，其中包含有关提交数据块的事务中的语句的信息。此结构具有以下字段：

- statements — PartiQL 语句列表以及它们何时开始运行的startTime。每个语句都有一个 statementDigest 哈希，这是计算 transactionInfo 结构哈希值所必需的。
- documents — 由语句更新的文档 ID。每个文档都包括它所属的tableName 和 tableId，以及更新该文档的每条语句的索引。

revisions

数据块中提交的文档修订列表。每个修订版本结构都包含所有此版本的所有提交视图字段。

这也可以包括代表仅限内部的系统修订版哈希，这些版本是日记账账完整哈希链的一部分。

已编校的修订版

在 Amazon QLDB，DELETE 语句只能通过创建将文档标记为已删除的新修订版，从逻辑上删除文档。QLDB 还支持数据编校操作，允许您永久删除表历史记录中的非活动文档修订版本。

编校操作仅删除指定修订版中的用户数据，而日记账序列和文档元数据则保持不变。这样可保持分类账的整体数据完整性。有关编校操作示例的更多信息，请参阅 对文档修订版执行编校。

已编校的修订示例

考虑前面的 数据块示例。在此数据块，假设您编辑了文档 ID 为HwVFkn8IMRa0xjze5xcgga、版本号为0的修订版。

编校完成后，修订版中的用户数据 (由data结构表示) 将替换为一个新 dataHash 字段。该字段的值是已移除 data 结构的 Ion 哈希。因此，分类账保持了其整体数据的完整性，并通过现有验证 API 操作保持加密可验证性。

以下修订示例显示了此密文结果，新dataHash字段以####突出显示。

> **Note**
>
> 本文档仅用于参考。显示的哈希值并非实际计算的哈希值。

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

QLDB 还会在日记账中为已完成编辑请求添加新数据块。此数据块还包含其他 redactionInfo 条目，其中包含在事务中已编辑的修订的列表，如以下示例所示。

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

```
      version:0
    }
  ]
}
...
```

示例应用程序

有关使用导出数据验证日记账哈希链的 Java 代码示例，请参阅 GitHub 存储库 aws-samples/amazon-qldb-dmv-sample-java。此示例应用程序包含以下类别文件：

- ValidateQldbHashChain.java - 包含从分类账中导出日记账账数据块、并使用导出的数据验证数据块之间哈希链的教程代码。
- JournalBlock.java - 包含名为`verifyBlockHash()`的方法，该方法演示如何计算数据块中的每个哈希分量。此方法由`ValidateQldbHashChain.java`中的教程代码调用。

有关如何下载和安装此完整示例应用程序的说明，请参阅 安装 Amazon QLDB Java 示例应用程序。在运行教程代码之前，请确保按照Java 教程中的步骤 1—3 设置示例分类账并向其中加载示例数据。

另请参阅

有关 QLDB 中的日记账的更多信息，请参阅以下主题：

- 从Amazon QLDB 导出日记账数据 — 了解如何将日记账数据导出至 Amazon Simple Storage Service (Amazon S3)。
- Amazon QLDB 流式传输日记账数据— 了解如何将日记账数据流式传输至 Amazon Kinesis Data Streams。
- Amazon QLDB 中的数据验证 — 了解日记账数据的加密验证。

Amazon QLDB 词汇表

以下是您在使用 Amazon QLDB 时可能遇到的关键术语的定义。

数据块 | 摘要 | 文档 | 文档 ID | 文档修订 | 条目 | field | index | 索引存储 | 日记账 | 日记账块 | 日记账存储 | 日记账链 | 日记账小贴士 | 分类账 | 证明 | 修订 | session | 链 | table | 表视图 | 查看

数据块

在事务中提交到日记账的对象。单个事务在日记账中写入一个块，因此一个块只能与一个事务相关联。数据块包含代表事务中提交的文档修订版本条目和提交这些修订的 PartiQL 语句。

每个块也有一个用于验证的哈希值。块哈希值是根据该块内的条目哈希值与前一个链块的哈希值相结合计算出的。

摘要

一个 256 位的哈希值，它唯一地表示分类账截至某个时间点的整个文档修订历史记录。摘要哈希值是根据您的日记账中截至当时最新提交的块的完整哈希链计算得出的。

QLDB 允许您将摘要生成作为安全输出文件的形式。然后，您可以使用该输出文件来验证文档修订版本相对于该哈希值的完整性。

文档

一组 Amazon Ion `struct`格式的数据，可以在表中插入、更新和删除。QLDB 文档可以包含结构化、半结构化、嵌套和无架构数据。

文档 ID

QLDB 分配给插入到表中的每个文档的通用唯一标识符 (UUID)。此 ID 是一个 128 位的数字，它以 Base62 编码的字母数字字符串表示，固定长度为 22 个字符。

文档修订

一种表示由唯一文档 ID 标识的文档序列的单个版本的 Ion 结构。修订版既包括您的用户数据（即您在表中写入的数据），也包括系统生成的元数据。每个修订版都与表相关联，并由文档 ID 和从零开始的版本号组合作为唯一标识。

条目

块中包含的对象。条目表示在事务中插入、更新和删除的文档修订以及提交这些修订的 PartiQL 语句。

每个条目还有一个用于验证的哈希值。条目哈希值是根据该条目中的修订哈希值或语句哈希值计算得出的。

field

构成 QLDB 文档每个属性的名称-值对。名称为符号令牌，其值不受限制。

index

可以在表上创建的一种数据结构，用于优化数据检索操作的性能。有关 QLDB 中索引的信息，请参阅 Amazon QLDB PartiQL 参考中的 CREATE INDEX。

索引存储

分类账的表、索引和索引历史记录使用的磁盘空间。索引存储包含的分类账数据针对高性能查询进行了优化。

日记账

在分类账中提交的所有块的哈希链集。日记账仅用于追加，它代表了对分类账数据的所有更改的、完整且不可变的历史记录。

日记账块

请参阅数据块。

日记账存储

分类账的日记账账使用的磁盘空间。

日记账链

请参阅链。

日记账小贴士

日记账在某一时间点上最近提交的块。

分类账

Amazon QLDB 分类账数据库资源的实例。这是 QLDB 中的主要 AWS 资源类型。分类账由日记账存储和索引存储组成。分类账数据提交到日记账后，即可在 Amazon Ion 文档修订版表格中进行查询。

证明

QLDB 为给定摘要和文档修订版返回的 256 位哈希值的有序列表。它由 Merkle 树模型将给定的修订散列链接到摘要散列所需的散列组成。您可以使用证明来验证您的修订版本相对于摘要的完整性。有关更多信息，请参阅Amazon QLDB 中的数据验证。

修订

请参阅文档修订。

session

一个对象，用于管理有关您的数据事务请求以及对分类账的响应的信息。活动的会话（正在积极运行事务的会话）表示与分类账的单个连接。QLDB 支持每个会话正在运行的事务。

链

日记账的分区。QLDB 目前仅支持单链日记账。

table

在分类账的日记账中提交的文件修订的无序集合的实体化视图。

表视图

表中数据的可查询子集，基于提交到日记账的事务。在 PartiQL 语句中，视图由表名的前缀限定符（以 `_ql_` 开头）表示。

您可以使用 `SELECT` 语句查询以下系统定义的视图：

- 用户 - 仅包含您在表中写入的数据的最新有效版本（即用户数据的当前状态）。这是 QLDB 中的默认视图。
- 已提交 – T您的用户数据和系统生成的元数据的最新有效版本。这是与您用户表直接对应的完整系统定义表。例如：`_ql_committed_`*`TableName`*。

查看

请参阅表视图。

访问 Amazon QLDB

您可以使用、 AWS Command Line Interface AWS CLI() 或 QLDB API AWS Management Console访问亚马逊 QLDB。下面各部分介绍了使用这些选项，以及使用的先决条件。

先决条件

在访问 QLDB 之前，如果尚未设置，则必须先 AWS 账户 进行设置。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

管理 IAM 中的 QLDB 权限

有关使用 AWS Identity and Access Management (IAM) 管理用户的 QLDB 权限的信息，请参阅。Amazon MQ 如何与 IAM 协同工作

授权以编程方式访问（可选）

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS Management Console。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的配置 AWS CLI 以使用。 • 有关 AWS 软件开发工具包、工具和 AWS API，请参阅《软件开发工具包和AWS 工具参考指南》中的 IAM 身份中心身份验证。
IAM	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用

哪个用户需要编程式访问权限？	目的	方式
		户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDK 和工具，请参阅 S AWS DK 和工具参考指南中的使用长期凭证进行身份验证。 • 有关 AWS API，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

如何访问Amazon QLDB

完成设置的先决条件后 AWS 账户，请参阅以下主题以了解有关如何访问 QLDB 的更多信息：

- 使用 控制台
- 使用 AWS CLI （仅限管理 API）
- 使用 Amazon QLDB Shell (仅限数据 API)
- 使用 API

通过控制台访问 Amazon QLDB

你可以通过 https://console.aws.amazon.com/qldb 访问亚马逊 QLDB 版。 AWS Management Console

您可以使用控制台在 QLDB 中执行以下操作：

- 创建、删除、描述和列出分类账。
- 使用PartiQL 编辑器运行PartiQL语句。
- 管理 QLDB 资源标签。
- 以加密方式验证日志数据。
- 导出或流式传输日志数据块。

要了解如何创建 Amazon QLDB 分类账和设置示例应用程序数据，请参阅 Amazon QLDB 控制台入门。

PartiQL 编辑器快速参考

Amazon QLDB 支持 PartiQL 的子集作为其查询语言，支持 Amazon Ion作为其文档数据格式。有关在 PartiQL 上实施 QLDB 的完整指南和更多详细信息，请参阅Amazon QLDB PartiQL 参考。

以下主题提供了有关如何在 QLDB 中使用 PartiQL 的快速参考概述。

主题

关于 QLDB 中 PartiQL 的快速小贴士

以下是在 QLDB 中使用 PartiQL 的提示和最佳实践小贴士：

- 了解并发和事务限制 — SELECT 查询等所有语句都应遵守乐观并发控制（OCC）冲突和事务限制，包括 30 秒事务暂停。
- 使用索引 - 使用高基数索引，并运行有针对性的查询来优化语句并避免全表扫描。要了解更多信息，请参阅优化查询性能。
- 使用相等谓词 - 索引查找需要相等运算符（= 或 IN）。不等式运算符（<、>、LIKE、BETWEEN）不符合索引查找的条件，因此会生成全表扫描。
- 仅使用内部联接 - QLDB 仅支持内部联接。根据最佳实践标准，在为要加入的每个表编制索引的字段上进行联接。为联接条件与相等谓词选择高基数索引。

命令

Amazon QLDB 支持以下 PartiQL 命令。

数据定义语言 (DDL)

命令	描述
CREATE INDEX	为表内的顶级文档字段创建索引。
CREATE TABLE	创建表。
DROP INDEX	从表中删除索引。
DROP TABLE	停用现有表。
取消删除表	重新激活非活动表。

数据操作语言（DML）

命令	描述
删除	通过创建文档最终修订版，将活动文档标记为已删除。
FROM（插入、删除或设置）	语义与UPDATE相同。
INSERT	向表格中添加一个或多个 文档。
SELECT	从一个或多个表中检索数据。
更新	更新、插入或移除文档中的特定元素。

DML 语句示例

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'
```

选择 - 关联子查询

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

选择 - 内部联接

```
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')
```

选择 - 使用 BY 子句获取文档 ID

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

System-defined 视图

QLDB 支持以下系统定义表视图。

查看	描述
table_name	表格默认用户视图，仅包含用户数据的当前状态。
_ql_committed_ *table_name*	表格的完整系统定义提交视图，其中包含用户数据和系统生成的元数据（例如文档 ID）的当前状态。
history(*table_name*)	内置历史记录函数，可返回表格的完整修订历史记录。

基本语法规则

QLDB 支持以下适用于 PartiQL 的基本语法规则。

字符	描述
'	在单引号表示字符串值或 Amazon Ion 结构的字段名称。
"	双引号表示带引号的标识符，如用作表名的 保留字。
`	反引号表示 Ion 文本值。
.	用点表示法访问父结构的字段名称。
[]	方括号定义 Ion `list`，或者表示现有列表从零开始的序数。
{}	大括号定义 Ion `struct`。
<< >>	双尖括号定义了 PartiQL 数据包，这是无序集合。你用一个数据包将多个文档插入表格。
区分大小写	所有 QLDB 系统对象名称（包括字段名、表名）均区分大小写。

PartiQL 编辑器键盘快捷键

QLDB 控制台上的 PartiQL 编辑器支持以下键盘快捷键。

操作	macOS	Windows
运行	Cmd+Return	Ctrl+Enter
注释	Cmd+/	Ctrl+/
Clear	Cmd+Shift+Delete	Ctrl+Shift+Delete

使用 AWS CLI （仅限管理 API）访问亚马逊 QLDB

您可以使用 AWS Command Line Interface (AWS CLI) AWS 服务 从命令行控制多个命令并通过脚本自动执行这些操作。您可以根据需要使用 AWS CLI 进行一次性操作。您还可以使用它在实用工具脚本中嵌入 Amazon QLDB 操作。

要进行 CLI 访问，您需要访问密钥 ID 和秘密访问密钥。如果可能，请使用临时凭证代替长期访问密钥。临时凭证包括访问密钥 ID、秘密访问密钥，以及一个指示凭证何时到期的安全令牌。有关更多信息，请参阅 IAM 用户指南中的将临时证书与 AWS 资源配合使用。

有关 AWS CLI中对 QLDB 可用的所有命令的完整列表和使用示例，请参阅 AWS CLI 命令参考。

> **Note**
>
> AWS CLI 仅支持中列出的`qldb`管理 API 操作Amazon QLDB API 参考。此 API 仅用于管理分类账资源和非事务数据操作。
> 要使用命令行界面通过 `qldb-session`API 运行数据事务，请参阅 使用 QLDB Shell 访问 Amazon QLDB (仅限数据 API)。

主题

- 安装和配置 AWS CLI
- AWS CLI 与 QLDB 一起使用

安装和配置 AWS CLI

它们可以在 Linux、macOS 或 Windows 上 AWS CLI 运行。要进行安装和配置，请参见 AWS Command Line Interface 用户指南 的以下说明：

1. 安装或更新最新版本的 AWS CLI
2. 快速设置

AWS CLI 与 QLDB 一起使用

命令行格式包含 Amazon QLDB 操作名称，后跟该操作的参数。除了 AWS CLI JSON 之外，还支持参数值的简写语法。

使用 `help` 帮助列出 QLDB 中的所有可用命令：

```
aws qldb help
```

您还可以使用 `help` 来描述特定命令并了解有关其用法的详细信息：

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

例如，要创建分类账，执行以下操作：

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

使用 QLDB Shell 访问Amazon QLDB (仅限数据 API)

Amazon QLDB 提供了一个命令行 Shell，用于与事务数据 API 进行交互。使用 QLDB Shell，您可以对分类账数据运行 PartiQL语句。

这个 Shell 的最新版本是用 Rust 编写的，在默认`main`分支上的 GitHub 存储库awslabs/amazon-qldb-shell中是开源的。Python 版本 (v1) 也仍然可以在 `master` 分支的同一个存储库中使用。

> **Note**
>
> Amazon QLDB Shell 仅支持`qldb-session`事务数据 API。此 API 仅用于在 QLDB 分类账上运行 PartiQL 语句。
> 要使用命令行界面与 `qldb` 管理 API 操作进行交互，请参阅 使用 AWS CLI （仅限管理 API）访问亚马逊 QLDB。

此工具不会整合至应用程序中或用于生产目的。该工具旨在让您快速试验 QLDB 和 PartiQL。

以下各节介绍如何开始使用 QLDB Shell。

主题

先决条件

开始使用 QLDB Shell之前，您必须执行以下操作：

1. 按照 访问 Amazon QLDB 中的 AWS 的设置说明进行操作。这包括以下这些：

   1. 注册AWS。
   2. 创建具有适当 QLDB 权限的用户。
   3. 授权以编程方式访问开发。
2. 设置您的AWS凭据和默认凭据AWS 区域。有关说明，请参阅 AWS Command Line Interface 用户指南中的配置基础。

   有关可用区域的完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。
3. 在分类账的 STANDARD 权限模式下，您必须创建 IAM 策略，以授予在适当表格上运行 PartiQL 语句的权限。要了解如何创建这些策略，请参阅 请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

安装 Shell

要安装 QLDB Shell 的最新版本，请参阅 GitHub 中的README.md。QLDB 在 GitHub 存储库的 发布部分提供了适用于 Linux、macOS 和 Windows 的预建二进制文件。

对于 macOS，Shell 与`aws/tap` Homebrew 选项卡集成。若要使用 Homebrew 在 macOS 上安装 Shell，请运行以下命令。

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

配置

安装后，Shell 会加载初始化 $XDG_CONFIG_HOME/qldbshell/config.ion 期间位于的默认配置文件。在 Linux 和 macOS，此文件通常位于 ~/.config/qldbshell/config.ion。如果这样的文件不存在，Shell 将按默认设置运行。

安装后可以手动创建 config.ion 文件。此配置文件采用 Amazon Ion 数据格式。以下是最小 config.ion 文件的示例。

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

如果配置文件中未设置default_ledger，则在调用 Shell 时必须使用--ledger参数。有关配置选项的完整列表，请参阅 GitHub 的 README.md 文件。

调用 Shell

要在命令行终端上为特定分类账调用 QLDB Shell，请运行以下命令。用你的分类账名称替换*my-example-ledger* 。

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

此命令连接至您的默认AWS 区域。要明确指定区域，您可以使用 --region 或 --qldb-session-endpoint 参数运行命令，如下一节所述。

调用 qldb Shell 会话后，您可输入以下类型的输入：

- Shell 命令
- 在单独的事务中采用单个 Partiql 语句
- 单个事务中包含多个 PartiQL 语句

Shell 参数

要查看调用 Shell 的可用标志和选项的完整列表，请按如下方式运行带有 --help 标志的 qldb 命令。

```
$ qldb --help
```

以下是 qldb 命令的一些键标和选项。您可以添加这些可选参数以覆盖AWS 区域、凭据配置文件、端点、结果格式和其他配置选项。

用量

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

FLAGS

-h, **--help**

打印帮助信息。

-v, **--verbose**

配置日志详细程度。默认情况下，Shell 程序仅记录错误。要提高详细程度，请重复此参数 (例如-vv)。最高等级为对应于trace的-vvv。

-V, **--version**

打印版本信息。

OPTIONS

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

要连接的分类账的名称。如果default_ledger未在您的config.ion文件中设置，则其为 Shell 必填参数。在此文件中，您可以设置其他选项，如区域。

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

可在其中定义任何 Shell 配置选项的文件。有关格式详细信息和配置选项的完整列表，请参见 GitHub 上的 README.md 文件。

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

查询结果的输出格式。默认为 ion。

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

用于身份验证的 AWS 凭据配置文件的位置。

如果未提供，Shell 可使用您的默认AWS配置文件，该配置文件位于~/.aws/credentials。

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

要连接的 QLDB 分类账AWS 区域代码。例如：us-east-1。

如果未提供，则 Shell 将连接至您的AWS配置文件中指定的默认AWS 区域。

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

用于连接的 qldb-session API 端点。

有关可用 QLDB 区域和端点的完整列表，请参阅 AWS 一般参考中的 Amazon S3 端点和限额。

命令参考

调用 qldb会话后，Shell 支持以下密钥和数据库命令：

Shell 秘钥

键	函数描述
Enter	运行语句。
Escape+Enter (macOS, Linux) Shift+Enter (Windows)	开始新行，输入一条跨越多行的语句。您也可以复制多行输入文本，然后将其粘贴至 Shell。 有关在 macOS 中设置Option而非Escape作为元密钥的说明，请参阅OS X Daily网站。
Ctrl+C	取消当前命令。
Ctrl+D	发出文件结束信号 (EOF) 并退出 Shell 的当前级别。如不在活动事务中，则退出 Shell。在活动事务中，请中止该事务。

Shell 数据库命令

命令	函数描述
help	显示帮助信息。

命令	函数描述
`begin`	开始事务。
`start transaction`	
`commit`	将您的事务提交至分类账日志账中。
`abort`	停止您的事务，并拒绝您所做的任何更改。
`exit`	退出 Shell。
`quit`	

> **Note**
>
> 所有 QLDB Shell 命令均不区分大小写。

运行单独语句

除了 README.md 中列出的数据库命令和 Shell 元命令外，Shell 程序会将您输入的每条命令解释为单独的 PartiQL 语句。默认情况下，Shell 会启用`auto-commit`模式。此模式为可配置。

在`auto-commit`模式下，Shell 会在自己的事务中隐式运行每条语句，如果未发现错误，则自动提交事务。这意味着您不必在每次运行语句时都手动运行`start transaction` (或 `begin`) 和 `commit`。

管理事务

或者，QLDB Shell 允许您手动控制事务。您可在事务中以交互方式运行多个语句，也可以通过按顺序批处理命令和语句来以非交互方式运行多个语句。

交互式事务

若要运行交互式事务，请执行以下步骤。

1. 要开始事务，请输入 `begin` 命令。

```
qldb> begin
```

开始事务后，Shell 将显示以下命令提示符。

```
qldb *>
```

2. 然后，您输入的每条事务都在同一笔事务中运行。
   - 例如，您可按以下方式运行单个语句。

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

     按下Enter后，Shell 会显示语句的结果。
   - 您也可以输入多个由分号 (;) 分隔符分隔的语句或者命令，如下所示。

     ```
     qldb *> SELECT * FROM Vehicle WHERE VIN = '1N4AL11D75C109151'; commit
     ```
3. 若要结束事务，请输入以下命令之一。
   - 输入commit命令将您的事务提交至分类账日志账中。

     ```
     qldb *> commit
     ```
   - 输入abort命令，以停止您的事务并拒绝您所做的任何更改。

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

事务超时限制

交互式事务遵守 QLDB 事务超时限制。如果您未在事务启动后的 30 秒内提交此事务，QLDB 会自动使该事务过期，并拒绝事务期间所做的任何更改。

然后，Shell 不显示语句结果，而是显示一条过期错误消息并返回到普通的命令提示符。要重试，必须再次输入 begin 命令才能开始新的事务。

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

非交互式事务

您可以按以下顺序对命令和语句进行批处理，从而运行包含多条语句的完整事务。

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

必须用分号 (;) 分隔符分隔每条命令与语句。如果事务中的任何语句无效，则 Shell 会自动拒绝该事务。Shell 不会继续处理您随后输入的任何语句项。

您还可设置多笔事务。

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

与前面的示例类似的是，如果事务失败，Shell 将不会继续处理您输入的任何后续事务或语句。

如果您不结束事务，Shell 会切换至交互模式，并提示您输入下一个命令或语句。

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

退出 Shell

要退出当前 qldb Shell 会话，请输入`exit`或`quit`命令，或者在 Shell 不在事务中时使用键盘快捷键Ctrl+D。

```
qldb> exit
$
```

```
qldb> quit
$
```

示例

有关在 QLDB 中编写 PartiQL 语句的信息，请参阅Amazon QLDB PartiQL 参考。

Example

以下示例显示基本命令常见序列。

> **Note**
>
> QLDB Shell 程序在自己的事务中运行本示例中的每条 PartiQL 语句。

此示例假设分类账 `test-ledger` 已经存在且处于活动状态。

```
$ 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
```

使用 API 访问 Amazon QLDB

您可以使用 AWS Management Console 和 AWS Command Line Interface (AWS CLI) 与 Amazon QLDB 进行交互式协作。但是，要充分利用 QLDB，您可以使用 QLDB 驱动程序或 AWS SDK 编写应用程序代码，以便使用 API 与账本进行交互。

该驱动程序允许您通过事务数据 API 与 QLDB 交互。该 AWS 软件开发工具包支持与 QLDB 资源管理 API 的交互。有关这些 API 的更多信息，请参阅 Amazon QLDB API 参考。

此驱动程序对Java、.NET、Go、Node.js和 Python 中的 QLDB 提供支持。要快速开始使用这些语言，请参阅 Amazon QLDB 驱动程序入门。

必须先授予编程访问权限，然后才能在应用程序中使用 QLDB 驱动程序或 AWS SDK。有关更多信息，请参阅 授予编程式访问权限。

Amazon QLDB 控制台入门

本教程指导您完成创建第一个 Amazon QLDB 分类账的步骤，并用表和样例数据填充它。您在此场景中创建的示例分类账是一个机动车辆部门 (DMV) 数据库，用于追踪有关车辆登记的完整历史信息。

资产的历史是 QLDB 的常见用例，因为它涉及各种场景和操作，这些场景和操作突显了分类账数据库的用处。使用 QLDB，您可以在支持类似 SQL 的查询功能的面向文档的数据库中直接访问、查询和验证数据更改的完整历史记录。

使用本教程，以下主题说明了如何添加车辆登记、修改车辆登记，以及如何查看这些登记的更改历史记录。本指南还向您展示了如何以加密方式验证注册文档，并通过清理资源和删除示例分类账来结束。

本教程中的每个步骤都有使用的 AWS Management Console 说明。

主题

教程先决条件和注意事项

在开始本 Amazon QLDB 教程之前，请确保满足以下先决条件：

1. 如果您尚不了解，请按照 访问 Amazon QLDB 中的 AWS 设置说明进行操作。这些步骤包括注册 AWS 和创建管理用户。
2. 按照中的 设置权限 说明为您的 QLDB 资源设置 IAM 权限。要完成本教程中的所有步骤，您需要通过 AWS Management Console 对分类账资源拥有完全管理权限。

> **Note**
>
> 如果您已以具有完整 AWS 管理权限的用户身份登录，可以跳过此步骤。

3. （可选）QLDB 使用 AWS Key Management Service (AWS KMS) 中的密钥对静态数据进行加密。选择以下 AWS KMS keys 类型之一：

   - AWS 拥有的 KMS 密钥 – 使用 AWS 代表您拥有和管理的 KMS 密钥。这是默认选项，无需额外设置。
   - 客户托管的 KMS 密钥 – 在您创建、拥有和管理的账户中使用指定的对称加密 KMS 密钥。QLDB 不支持非对称密钥。

     此选项要求您创建 KMS 密钥或使用账户中现有密钥。有关创建客户托管密钥的说明，请参阅 AWS Key Management Service《开发者指南》 中的 创建对称加密 KMS 密钥。

     要指定客户托管的 KMS 密钥，您可以使用其密钥 ID、别名或 Amazon 资源名称(ARN)。有关更多信息，请参阅 AWS Key Management Service《开发者指南》 中的 密钥标识符 (KeyId)。

     > **Note**
     >
     > 不支持跨区域密钥。指定的 KMS 密钥必须与您的分类账处于同一 AWS 区域 中。

设置权限

在此步骤中，您将通过控制台为 AWS 账户 中的所有 QLDB 资源设置完全访问权限。要快速授予这些权限，请使用 AWS 托管策略 AmazonQLDBConsoleFullAccess。

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和组：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中 创建权限集 的说明进行操作。
- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：

- 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
- （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

> **Important**
>
> 在本教程中，您授予自己对所有 QLDB 资源的完全管理权限。但是，对于生产用例，请遵循授予最低权限，或仅授予执行任务所需的权限这一安全最佳实践。有关示例，请参阅 Amazon QLDB 基于身份的策略示例。

要创建名为 `vehicle-registration` 的分类账，请进入 第 1 步：创建新分类账。

第 1 步：创建新分类账

在此步骤中，您将创建一个名为 `vehicle-registration` 的新 Amazon QLDB 分类账。然后，确认分类账的状态为 Active。您还可以验证添加到分类账中的任何标签。

创建分类账时，将默认启用 删除保护。QLDB 中有一项删除保护功能，可防止分类账被任何用户删除。在使用 QLDB API 或 AWS Command Line Interface (AWS CLI) 创建分类账时，您可以禁用删除保护。

创建新的分类账

1. 登录 AWS Management Console 并打开 Amazon QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 开始。
3. 在创建您的第一张分类账明细中，选择创建分类账。
4. 在 穿件分类账 页面上，执行以下操作：
   - 分类账信息 - 分类账名称应预先填充。**`vehicle-registration`**
   - 权限模式 - 要分配给分类账的权限模式。请选择以下任一选项：
     - 允许所有 – 这是一种旧式权限模式，支持对分类账进行 API 级别粒度的访问控制。

       此模式允许拥有此分类账的 `SendCommand` API 权限的用户在指定分类账中的任何表上运行所有 PartiQL 命令（因此，`ALLOW_ALL`）。此模式忽略您为分类账创建的任何表级或命令级 IAM 权限策略。

- 标准 – (推荐) 一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。我们强烈建议使用此权限模式来最大限度地提高分类账数据的安全性。

  默认情况下，此模式拒绝所有用户请求在此分类账中的任何表上运行任何 PartiQL 命令。要允许 PartiQL 命令运行，除了分类账的 SendCommand API 权限外，您还必须为特定表资源和 PartiQL 操作创建 IAM 权限策略。有关信息，请参阅 请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

- 加密静态数据 – 用于在 AWS Key Management Service (AWS KMS) 中的加密静态数据的密钥。请选择以下任一选项：
  - 使用AWS拥有的 KMS 密钥 – 使用 AWS 代表您拥有和管理的 KMS 密钥。这是默认选项，无需额外设置。
  - 选择不同的 AWS KMS 密钥 – 在您创建、拥有和管理的账户中使用指定的对称加密 KMS 密钥。

    要创建新密钥，使用 AWS KMS 控制台，选择创建 AWS KMS 密钥。有关更多信息，请参阅 AWS Key Management Service开发人员指南 中的 创建对称加密 KMS 密钥。

    要使用现有 KMS 密钥，请从下拉列表中选择一个密钥或指定 KMS 密钥 ARN。
- 标签 – (可选) 通过以键值对的形式附加标签来向分类账添加元数据。您可以向分类账添加标签来帮助组织和标识它们。有关更多信息，请参阅为 Amazon QLDB 资源贴标签。

  选择添加标签，然后输入合适的键值对。

5. 根据需要进行设置后，选择 创建分类账。

> **Note**
>
> 当 QLDB 分类账的状态变为“活跃”时，您可以访问该 QLDB 分类账。这个过程可能需要几分钟。

6. 在分类账列表中，找到 vehicle-registration 并确认该分类账的状态是否为 Active。
7. （可选）选择 vehicle-registration 分类账名称。在车辆登记分类账详细信息页面上，确认您添加到分类账的所有标签都显示在标签卡上。您也可以使用此控制台页面编辑分类账标签。

要在 vehicle-registration 分类账中创建表，请继续 步骤 2：在分类账中创建表、索引和示例数据。

步骤 2：在分类账中创建表、索引和示例数据

当您的 Amazon QLDB 分类账处于活动状态时，您可以开始创建有关车辆、车主和注册信息的数据表。创建表和索引后，可以向其中加载数据。

在此步骤中，您将在`vehicle-registration`分类账中创建四个表格：

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

您还可创建以下索引。

表名称	Field
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicensePlateNumber`
`DriversLicense`	`PersonId`

您可以使用 QLDB 控制台自动创建这些带有索引的表，并在其中加载示例数据。或者，您可以使用控制台上的 PartiQL 编辑器逐步手动运行每个 PartiQL 语句。

自动选项

创建表、索引与示例数据

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 开始。
3. 在示例应用程序数据卡上的 自动选项下，在分类账列表中进行选择 `vehicle-registration`。

4. 选择 加在示例数据。

   如果操作成功完成，控制台将显示消息样本数据已加载。

   此脚本在单个事务中运行所有语句。如果事务的任何部分失败，则会回滚每条语句，并显示相应的错误消息。解决任何问题后，您可以重试该操作。

   > **Note**
   >
   > - 事务失败的一个可能原因是尝试创建重复表。如果您的分类账中已存在以下任何表名，则您的加载示例数据的请求将失败：`VehicleRegistration`、`Vehicle`、`Person`、和`DriversLicense`。
   >
   >   相反，请尝试将此示例数据加载到一个空分类账中。
   > - 此脚本运行参数化 `INSERT` 语句。因此，这些 PartiQL 语句使用绑定参数而不是文字数据记录在日记账块中。例如，您可能会在日记块中看到以下语句，其中问号 (`?`) 是文档内容的变量占位符。
   >
   >   ```
   >   INSERT INTO Vehicle ?
   >   ```

手动选项

使用空 `PrimaryOwner` 字段将文档插入 `VehicleRegistration`，使用空 `PersonId` 字段将文档插入 `DriversLicense`。稍后，您将使用 `Person` 表中系统分配的文档 `id` 填充这些字段。

> **Tip**
>
> 最佳做法是使用此文档 `id` 元数据字段作为外键。有关更多信息，请参阅查询文档元数据。

创建表、索引与示例数据

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 PartiQL 编辑器。
3. 选择 `vehicle-registration` 分类账。
4. 首先创建四个表。QLDB 支持开放内容且不强制架构，因此不需要定义属性或数据类型。

在查询编辑器窗口中输入以下语句，然后选择 运行。要运行语句，您可对 Windows 使用快捷键Ctrl+Enter，对 macOS 使用Cmd+Return。有关更多键盘快捷键的信息，请参阅 PartiQL 编辑器键盘快捷键。

```
CREATE TABLE VehicleRegistration
```

对以下每个步骤重复此步骤。

```
CREATE TABLE Vehicle
```

```
CREATE TABLE Person
```

```
CREATE TABLE DriversLicense
```

5. 接下来，创建可优化每个表的查询性能的索引。

> **Important**
>
> QLDB 需要索引才能高效查找文档。如果没有索引，QLDB 在读取文档时需进行全表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (=或IN) 运行带有WHERE谓词子句的语句。有关更多信息，请参阅优化查询性能。

在查询编辑器窗口中输入以下语句，然后选择 运行。

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

对以下每个步骤重复此步骤。

```
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. 创建索引后，您可以开始将数据加载到表中。在此步骤中，将包含分类账所跟踪车辆所有者的个人信息的文档插入 Person 表中。

   在查询编辑器窗口中输入以下语句，然后选择 运行。

```
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. 然后，在 DriversLicense 表格中填入包含每位车主驾驶执照信息的文档。

   在查询编辑器窗口中输入以下语句，然后选择 运行。

```
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. 现在，在 VehicleRegistration 表中填入车辆登记文件。这些文档包括存储主要和次要所有者的嵌套 Owners 结构。

   在查询编辑器窗口中输入以下语句，然后选择 运行。

```
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. 最后，在 Vehicle 表中填入描述在分类账中注册的车辆的文档。

   在查询编辑器窗口中输入以下语句，然后选择 运行。

```
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'
} >>
```

接下来，您可以使用 SELECT 语句从 vehicle-registration 分类账中的表中读取数据。继续执行第 3 步：查询分类账中的表。

第 3 步：查询分类账中的表

在 Amazon QLDB 分类账中创建表格和加载数据后，您可运行查询以查看刚刚插入的车辆登记数据。QLDB 使用 PartiQL作为其查询语言，使用 Amazon Ion作为其面向文档的数据模型。

PartiQL 是开源、与 SQL 兼容的查询语言，现已扩展为可与 Ion 配合使用。使用 PartiQL，您可使用熟悉的 SQL 运算符插入、查询和管理数据。Amazon Ion 是 JSON 的超集。Ion 是基于文档的开源数据格式，可让您灵活地存储和处理结构化、半结构化和嵌套数据。

在此步骤中，您将使用 `SELECT` 语句从 `vehicle-registration`分类账中的表中读取数据。

> **Warning**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。PartiQL 之所以支持此类查询，是因为其与 SQL 兼容。但是，切勿在 QLDB 中对生产用例运行表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`) 运行带有`WHERE`谓词子句的语句。有关更多信息，请参阅优化查询性能。

查询表格

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 PartiQL 编辑器。
3. 选择 `vehicle-registration` 分类账。
4. 在查询编辑器窗口中，输入以下语句在 `Vehicle` 表中查询您添加到分类账中的特定车辆识别码 (VIN)，然后选择 Run（运行）。

   要运行语句，您可对 Windows 使用快捷键Ctrl+Enter，对 macOS 使用Cmd+Return。有关更多键盘快捷键的信息，请参阅 PartiQL 编辑器键盘快捷键。

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

5. 您也可编写内部联接查询。此查询示例连接 `Vehicle` 与 `VehicleRegistration`，并返回注册信息以及指定 `VIN` 的已注册车辆的属性。

   输入以下语句并选择运行。

```
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
```

您也可以加入 Person 和 DriversLicense 表来查看与添加到分类账的驱动程序相关的属性。

对以下每个步骤重复此步骤。

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

要了解如何修改 vehicle-registration 分类账表格中的文档，请参阅 第 4 步：修改分类账中的文档。

第 4 步：修改分类账中的文档

现在您有了要处理的数据，可以开始在 Amazon QLDB 中对 vehicle-registration 分类账文档进行更改。例如，以带VIN 1N4AL11D75C109151 的奥迪 A5 为例。这辆车最初属于华盛顿州西雅图的一位名叫 Raul Lewis 的司机。

假设 Raul 把车卖给了华盛顿州埃弗雷特的一位名叫 Brent Logan 的居民。然后，Brent和 Alexis Pena 决定结婚。Brent 想把 Alexis 列为第二车主。在此步骤中，以下数据操作语言 (DML) 语句演示了如何在分类账中进行适当的更改以反映这些事件。

> **Tip**
>
> 最佳做法是使用系统分配的文档 id 作为外键。虽然您可以定义作为唯一标识符的字段（例如车辆的 VIN），但文档的真正唯一标识符是id。字段都包含在文档元数据中，您可以在提交视图（系统定义的表格视图）中对其进行查询。
> 有关 QLDB 中的视图的更多信息，请参阅核心概念。了解有关元数据的更多信息，请参阅 查询文档元数据。

修改文档

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。

2. 在导航窗格中，选择 PartiQL 编辑器。
3. 选择 vehicle-registration 分类账。

> **Note**
>
> 如果您使用控制台的自动加载样本数据功能设置分类账，请跳至步骤 6。

4. 如果您手动运行 INSERT 语句来加载示例数据，请继续执行这些步骤。

   要最初将 Raul 注册为这辆车的车主，首先要在 Person表中找到他由系统分配的文件 id。字段都包含在文档元数据中，您可以在提交视图（系统定义的表格视图）中对其进行查询。

   在查询编辑器窗口中输入以下语句，然后选择 运行。

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

   前缀_ql_committed_是保留的前缀，表示您要查询的Person 表的已提交视图。在此视图中，您的数据嵌套至data字段中，元数据嵌套至metadata字段中。

5. 现在，在 UPDATE 语句中使用这个 id 来修改VehicleRegistration表中的相应文档。输入以下语句并选择运行。

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

   发布此语句，确认您修改了 Owners 字段。

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

6. 要将车辆的所有权转让给埃弗里特市的 Brent，请先使用以下语句从 Person 表中找到他的 id。

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

   接下来，使用这个 id 来更新 VehicleRegistration 表中的 PrimaryOwner 和 City。

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

发布此语句，确认您修改了 PrimaryOwner 和 City 字段。

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

7. 要将 Alexis 添加为汽车的次要车主，请找到她的 Person id。

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

然后，使用以下 FROM-INSERT DML 语句将这个 id 插入到 SecondaryOwners 列表中。

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

发布此语句，确认您修改了 SecondaryOwners。

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

若要查看 vehicle-registration 分类账中的这些更改，请参阅第 5 步：查看文档修订历史记录。

第 5 步：查看文档修订历史记录

在上一步中修改车辆 VIN 1N4AL11D75C109151 注册数据后，您可查询其所有注册车主的历史记录以及任何其他更新的字段。通过查询内置的 历史记录函数，您可以查看之前在中插入、更新和删除的车辆登记文件的所有三个修订版。

历史函数从表的已提交视图返回修订，其中包括应用程序数据和相关元数据。元数据准确显示了每次修订的时间、顺序以及提交修订的事务。

在此步骤中，您将在vehicle-registration分类账的VehicleRegistration表格中查询文档的修订历史记录。

若要查看修订历史记录

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 PartiQL 编辑器。
3. 选择 vehicle-registration 分类账。
4. 要查询文档的历史记录，首先要找到其唯一的 id。除了查询已提交的视图外，获取文档 id 的另一种方法是在表格的默认用户视图中使用 BY 关键字。要了解更多信息，请参阅 通过 BY 子句查询文档 ID。

   在查询编辑器窗口中输入以下语句，然后选择 运行。

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

5. 接下来，您可以使用此 id 值来查询历史记录函数。输入以下语句并选择运行。确保将这些 id 值替换为自己文档 ID 的适当值。

   ```
   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**
   >
   > 在本教程中，此历史查询将返回文档 ID ADR2LQq48kB9neZDupQrMm 的所有修订版本。最佳做法是，使用日期范围（开始时间和结束时间）和文档 ID 来限定历史查询。QLDB 中的每个 SELECT 查询都是在事务中处理的，并且受事务超时限制的约束。包含开始时间和结束时间的历史记录查询将从日期范围限定中获得便利。有关更多信息，请参阅历史记录函数。

   历史函数以与提交视图相同的模式返回文档。此示例投射您修改后的车辆登记数据。该输出值应该类似于以下内容。

VIN	City（城市）	所有者
"1N4AL11D 75C109151"	"Seattle"	{PrimaryOwner:{PersonId:""} ,SecondaryOwners:[]}
"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**
>
> 历史查询可能并不总是按顺序返回文档修订版。

查看输出并确认更改是否反映了您在 第 4 步：修改分类账中的文档 中所做的事情。

6. 然后，您可以检查每个修订版的文档元数据。输入以下语句并选择运行。同样，请务必根据需要将 `id` 值替换为您自己的证件 ID。

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

该输出值应该类似于以下内容。

versio	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"

这些元数据字段提供了有关每项何时修改以及由哪个事务修改的详细信息。从这些数据中，您可看到以下内容：

- 该文档由系统分配的id唯一标识：ADR2LQq48kB9neZDupQrMm。这是以 Base62 编码的字符串表示的通用唯一识别码 (UUID)。
- txTime 显示文档的初始修订版（版本0）是在 2019-05-23T19:20:360d-3Z 创建的。
- 每个后续事务会创建包含相同文档 id 和递增版本号的新修订版，以及更新的 txId和 txTime。

要以加密方式验证 vehicle-registration 分类账中的文档修订版，请继续第 6 步：验证分类账中的文档。

第 6 步：验证分类账中的文档

借助 Amazon QLDB，您可在 SHA-256 中使用加密哈希，从而有效地验证分类账日记账中文档的完整性。在此示例中，Alexis 和 Brent 决定通过在汽车经销商处用 VIN 1N4AL11D75C109151 交换车辆来升级到一款新车型。经销商通过向登记处核实车辆的所有权来开始这一流程。

要详细了解验证和加密哈希在 QLDB 中的工作原理，请参阅 Amazon QLDB 中的数据验证。

在此步骤中，您将验证 vehicle-registration 分类账中的文档修订版本。首先，您请求一份摘要，该摘要作为输出文件返回，并作为分类账整个变更历史记录的签名。然后，您要求提供与此摘要相关的修订证明。使用此证明，如果所有验证检查都可通过，可以验证修订版的完整性。

请求摘要

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择分类账。
3. 在分类账列表中，选择 vehicle-registration。
4. 选择“获取摘要”。获取摘要对话框显示以下摘要详细信息：
   - 摘要 - 您请求的摘要的 SHA-256 哈希值。
   - 摘要提示地址 - 您请求的摘要所涵盖的日记中的最新块位置。地址包含以下两个字段：
     - strandId — 包含数据块的日记账链的唯一 ID。
     - sequenceNo — 一个索引号，用于指定数据块在链中的位置。
   - 分类账 - 您请求摘要的分类账名称。
   - 日期 - 您请求摘要时的时间戳。
5. 检查摘要信息。然后选择 Save（保存）。您可以保留默认文件名，或输入新名称。

   此步骤将保存一个内容为 Amazon Ion 格式的纯文本文件。该文件的文件扩展名为 .ion.txt，包含前面对话框中列出的所有摘要信息。以下是摘要内容的示例。字段的顺序可能因您的浏览器而异。

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

6. 将此文件保存在以后可以访问的某个位置。在以下步骤中，您将使用此文件来验证文档修订版。

保存分类账摘要后，您可以开始根据该摘要验证文档修订的过程。

> **Note**
>
> 在用于验证的生产用例中，您可以使用先前保存的摘要，而不是连续执行这两个任务。最佳做法是，在日记账中写入要稍后验证的修订版本后，立即请求并保存摘要。

验证文档的修订版

1. 首先，在分类账中查询要验证的文档修订版本的 `id`和 `blockAddress`。字段都包含在文档元数据中，您可以在提交视图中对其进行查询。

   文档 `id` 是系统分配的唯一 ID 字符串。`blockAddress` 是一种 Ion 结构，用于指定提交修订版本的块位置。

   在 QLDB 控制台的导航窗格中，选择 PartiQL 编辑器。
2. 选择 `vehicle-registration` 分类账。
3. 在查询编辑器窗口中输入以下语句，然后选择 运行。

   ```
   SELECT r.metadata.id, r.blockAddress
   FROM _ql_committed_VehicleRegistration AS r
   WHERE r.data.VIN = '1N4AL11D75C109151'
   ```
4. 复制并保存查询返回的 `id` 和 `blockAddress` 值。请务必省略 `id` 字段的双引号。在 Amazon Ion 中，字符串数据类型用双引号分隔。
5. 您已经选择了文档修订版，现在可以开始对其进行验证。

   在导航窗格中选择验证。
6. 在“验证文档”表单中，在“指定要验证的文档”下，输入以下输入参数：

   - 分类账 - 选择 `vehicle-registration`。
   - 块地址 - 您在步骤 3 中查询返回的`blockAddress`值。
   - 文档 ID - 您在步骤 3 中查询返回的 `id` 值。
7. 在 “指定要验证的摘要” 下，通过选择 “选择摘要”，选择之前保存的摘要。如果文件有效，则会自动填充控制台上的所有摘要字段。或者，您可以直接从摘要文件中手动复制和粘贴以下值：

   - 摘要 - 摘要文件中的 `digest` 值。
   - 摘要提示地址 - 摘要文件中的 `digestTipAddress` 值。

8. 查看您的文档和摘要输入参数，然后选择Verify（验证）。

   控制台为您自动执行两个步骤：

   a. 向 QLDB 请求指定文档的证明。
   b. 使用 QLDB 返回的证明来调用客户端 API，该API会根据提供的摘要验证您的文档修订版本。

   控制台在验证结果卡中显示您的请求结果。有关更多信息，请参阅验证结果。
9. 要测试验证逻辑，请重复“验证文档修订版” 下的步骤 6—8，但要更改摘要输入字符串中的单个字符。这应该会导致您的验证请求失败，并显示相应的错误消息。

如果您不再需要使用vehicle-registration分类账，请继续步骤 7 (可选)：清除资源。

步骤 7 (可选)：清除资源

您可以继续使用 vehicle-registration 分类账。但是，如果您不再需要，请将其删除。

如果您的分类账启用了删除保护，您必须先禁用它，然后才能使用 QLDB API、AWS Command Line Interface (AWS CLI) 或 QLDB 删除分类账。

删除分类账

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择分类账。
3. 如果启用了分类账删除保护，则必须先禁用它。

   在分类账列表中，选择 vehicle-registration，然后选择编辑分类账。
4. 在编辑分类账页面上，关闭删除保护，然后选择确认更改。
5. 在分类账列表中，再次选择 vehicle-registration，然后选择删除。
6. 通过在提供的字段中输入 **delete vehicle-registration** 来确认这一点。

要了解有关在 QLDB 中使用分类账的更多信息，请参阅 Amazon QLDB 入门：后续步骤。

Amazon QLDB 入门：后续步骤

有关使用 Amazon QLDB 的更多信息，请参见以下主题：

- 在 Amazon QLDB 中处理数据与历史记录
- Amazon QLDB 驱动程序入门
- Amazon QLDB 并发模型
- 从Amazon QLDB 导出日记账数据
- Amazon QLDB 流式传输日记账数据
- Amazon QLDB 中的数据验证
- Amazon QLDB PartiQL 参考

Amazon QLDB 驱动程序入门

本节包含帮助您了解如何使用 Amazon QLDB 驱动程序开发 Amazon QLDB 的实践教程。该驱动程序建立在 AWS SDK 之上，该软件开发工具包支持与 QLDB API 的交互。

QLDB 会话抽象

该驱动程序在事务数据 API 中提供了高级抽象层 (QLDB Session)。它通过管理 SendCommand API 调用，简化了对分类账数据运行 PartiQL 语句的过程。这些 API 调用需要驱动程序为您处理的多项参数，包括会话管理、事务管理以及出现错误时的重试策略。该驱动程序还进行了性能优化，并采用了与 QLDB 交互的最佳实践。

> **Note**
>
> 要与 Amazon QLDB API 参考中列出的资源管理 API 操作进行交互，您可以直接使用 AWS SDK 而不是驱动程序。管理 API 仅用于管理分类账资源和非事务性数据操作，例如导出、流式传输和数据验证。

Amazon Ion 支持

此外，该驱动程序使用 Amazon Ion 库来支持在运行事务时处理 Ion 数据。这些库还负责计算 Ion 值的哈希值。QLDB 需要这些 Ion 哈希来检查数据事务请求的完整性。

驱动程序术语

该工具之所以被称为驱动程序，是因为它与其他提供开发者友好接口的数据库驱动程序相当。这些驱动程序同样封装了将一组标准命令和函数转换为服务的低级 API 所需的特定调用的逻辑。

该驱动程序在 GitHub 上是开源的，可用于以下编程语言：

- Java 驱动程序
- .NET 驱动程序
- Go 驱动程序
- Node.js 驱动程序
- Python 驱动程序

有关所有支持的编程语言的一般驱动程序信息以及其他教程，请参阅以下主题：

- 驱动程序会话管理
- 驱动程序推荐
- 驱动程序重试策略
- 常见错误
- 应用程序教程示例
- 使用 Amazon Ion
- 获取 PartiQL 语句统计信息

适用于 Java 的 Amazon QLDB 驱动程序

要处理分类账中的数据，您可以使用AWS提供的驱动程序从 Java 应用程序连接到 Amazon QLDB。以下主题介绍了如何开始使用 Java 上 QLDB 驱动程序。

主题

- 驱动程序资源
- 先决条件
- 设置您的默认 AWS 凭证和区域
- 安装
- 适用于 Java 的 Amazon QLDB 驱动程序 — 快速入门教程
- Java 的 Amazon QLDB 驱动程序 — 说明书参考

驱动程序资源

有关 Java 驱动程序支持功能的更多信息，请参阅以下资源：

- API 参考：2.x, 1.x
- 驱动程序源代码 (GitHub)
- 示例应用程序源代码 (GitHub)
- 分类账加载器框架 (GitHub)
- Amazon Ion 代码示例

先决条件

开始使用适用于 Java 的 QLDB 驱动程序之前，您必须执行以下操作:

1. 按照 访问 Amazon QLDB 中的 AWS 的设置说明进行操作。这包括以下这些：
    1. 注册AWS。
    2. 创建具有适当 QLDB 权限的用户。
    3. 授权以编程方式访问开发。
2. 通过下载并安装以下内容来设置一个 Java 开发环境：
    1. Java SE Development Kit 8，例如 Amazon Corretto 8。
    2. （可选）您选择的 Java 集成式开发环境 (IDE)，例如 Eclipse 或 IntelliJ。
3. 通过 设置您的默认 AWS 凭证和区域 为 AWS SDK for Java 配置您的开发环境。

接下来，您可下载完整的教程示例应用程序，也可以只在 Java 项目中安装驱动程序并运行短代码示例。

- 要在现有项目中安装 QLDB 驱动程序和 AWS SDK for Java，请继续执行安装。
- 要设置项目并运行演示分类账上基本数据事务的简短代码示例，请参阅 快速入门教程。
- 要在完整的教程示例应用程序中运行更深入的数据和管理 API 操作示例，请参阅 Java 教程。

设置您的默认 AWS 凭证和区域

QLDB 驱动程序和底层 AWS SDK for Java 要求在运行时向应用程序提供 AWS 凭证。本指南中的代码示例假设您使用 AWS 凭证文件，如 AWS SDK for Java 2.x《开发者指南》 中的 设置默认凭证和区域 所述。

作为这些步骤的一部分，您还应设置默认值 AWS 区域 以确定默认 QLDB 终端节点。在默认的 AWS 区域 中，代码示例连接到 QLDB。有关 QLDB 区域的完整列表，请参阅 AWS 一般参考中的 Amazon QLDB 端点和限额。

下面是一个名为 `~/.aws/credentials` 的 AWS 凭证文件示例，其中波浪号字符 (~) 表示主目录。

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

用您自己的 AWS 凭证值替换值 *your_access_key_id* 和 *your_secret_access_key*。

安装

QLDB 支持以下 Java 驱动程序版本及其 AWS SDK 依赖项。

驱动程序版本	AWS SDK	状态	最新发布日期
1.x	AWS SDK for Java 1.x	量产版	2023 年 3 月 20 日
2.x	AWS SDK for Java 2.x	量产版	2021 年 6 月 4 日

要安装 QLDB 驱动程序，我们建议使用依赖项管理系统，比如 Gradle 或 Maven。例如，将以下构件作为依赖项添加到您的 Java 项目中。

2.x

Gradle

在您的 `build.gradle` 配置文件中添加此依赖项。

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

Maven

在您的 `pom.xml` 配置文件中添加此依赖项。

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

此构件自动包含 AWS SDK for Java 2.x 核心模块、Amazon Ion 库和其他必需的依赖项。

1.x

Gradle

在您的 build.gradle 配置文件中添加此依赖项。

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

Maven

在您的 pom.xml 配置文件中添加此依赖项。

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

此构件自动包含 AWS SDK for Java 核心模块、Amazon Ion 库和其他必需的依赖项。

> **Important**
>
> Amazon Ion 命名空间 - 在应用程序中导入 Amazon Ion 类时，必须使用命名空间 com.amazon.ion 下的软件包。AWS SDK for Java依赖于software.amazon.ion命名空间下的另一个 Ion 包，但这是一个与 QLDB 驱动程序不兼容的旧包。

有关如何在分类账上运行基本数据事务的简短代码示例，请参阅 说明书参考。

其他可选库

另外，您还可以在项目中添加以下有用的库。这些构件是 Java 教程 示例应用程序中必需的依赖项。

1. aws-java-sdk-qldb - AWS SDK for Java 的 QLDB 模块。QLDB 支持的最低版本为 1.11.785。

   在您的应用程序中使用此模块可以直接与 Amazon QLDB API 参考 中列出的管理 API 操作进行交互。

2. jackson-dataformat-ion - 用于 Ion 的 FasterXML 的 Jackson 数据格式模块。示例应用程序需要版本 `2.10.0` 或更高版本。

Gradle

将这些依赖项添加到您的 `build.gradle` 配置文件中。

```
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

将这些依赖项添加到您的 `pom.xml` 配置文件中。

```
<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>
```

适用于 Java 的 Amazon QLDB 驱动程序 — 快速入门教程

在本教程中，您将学习如何使用适用于 Java 的最新版 Amazon QLDB 驱动程序来设置简单应用程序。本指南包括安装驱动程序的步骤以及创建、读取、更新和删除 (CRUD) 的基本操作的简短代码示例。有关在完整示例应用程序中演示这些操作的更深入的示例，请参阅 Java 教程。

主题

- 第 2 步：初始化驱动程序
- 第 3 步：创建表和索引
- 第 4 步：插入文档
- 第 5 步：查询文档
- 第 6 步：更新文档
- 运行完整的应用程序

先决条件

在开始之前，请务必执行以下操作：

1. 请为 Java 驱动程序完成 先决条件（如果尚未执行此操作）。这包括注册 AWS、授予开发所需的编程访问权限以及安装 Java 集成式开发环境（IDE）。
2. 创建一个名为 `quick-start` 分类账。

   要了解如何创建分类账，请参阅控制台入门中的 Amazon QLDB 分类账的基本操作 或 第 1 步：创建新分类账。

步骤 1：设置您的 项目

首先，您需要设置您的 Java 项目。我们建议在本教程中使用 Maven 依赖项管理系统。

> **Note**
>
> 如果您使用的 IDE 具有自动执行这些设置步骤的功能，则可以直接跳到 第 2 步：初始化驱动程序。

1. 为应用程序创建一个文件夹。

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

2. 输入以下命令从 Maven 模板初始化项目。用您自己的值替换 *project-package*、*project-name* 和 *maven-template*。

```
$ mvn archetype:generate
```

```
  -DgroupId=project-package \
  -DartifactId=project-name \
  -DarchetypeArtifactId=maven-template \
  -DinteractiveMode=false
```

对于 *maven-template*，你可以使用基本的 Maven 模板：maven-archetype-quickstart

3. 要将 Java 的 QLDB 驱动程序添加为项目依赖项，请导航到新创建的 pom.xml 文件并添加以下构件。

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

此构件自动包含 AWS SDK for Java 2.x 核心模块、Amazon Ion 库和其他必需的依赖项。您的 pom.xml 文件应类似于如下所示：

```
<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. 打开 App.java 文件。

   然后，按以下步骤中逐步添加代码示例，尝试一些基本的 CRUD 操作。或者，您可以跳过分步教程，改为运行完整的应用程序。

第 2 步：初始化驱动程序

初始化连接到名为 quick-start 的分类账的驱动程序实例。将以下代码添加到您的 App.java 文件。

```
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();
    }
}
```

第 3 步：创建表和索引

以下代码示例显示如何运行 CREATE TABLE 和 CREATE INDEX。

在 main 方法中，添加以下代码，创建名为 People 的表以及该表上的 lastName 字段的索引。索引是优化查询性能和帮助限制乐观并发控制 (OCC) 冲突异常所必需的。

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

第 4 步：插入文档

以下代码示例显示如何运行 INSERT 语句。QLDB 支持 PartiQL 查询语言（兼容 SQL）和 Amazon Ion 数据格式（JSON 的超集）。

添加以下代码，在 People 表格中插入文档。

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

此示例使用问号 (?) 作为变量占位符，将文档信息传递给语句。使用占位符时，必须传递一个 IonValue 类型的值。

> **Tip**
>
> 要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个 IonList 类型的参数（显示转换为 IonValue），如下所示。
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```
>
> 传递 IonList 列表时，不要将变量占位符 (?) 包括在双尖括号（<<...>>）内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

第 5 步：查询文档

以下代码示例显示如何运行 SELECT 语句。

添加以下代码，用于从 People 表格中查询文档。

```
// 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
});
```

第 6 步：更新文档

以下代码示例显示如何运行 UPDATE 语句。

1. 添加以下代码，通过更新 age 到 42 来更新 People 表中的文档。

   ```
   // 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. 再次查询文档以查看更新的值。

   ```
   // 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. 使用 Maven 或你的 IDE 来编译和运行该 App.java 文件。

运行完整的应用程序

以下代码示例是 App.java 应用程序的完整版本。您还可以从头到尾复制并运行此代码示例，而不必单独执行前面的步骤。此应用程序演示了对名为 quick-start 的分类账的一些基本的 CRUD 操作。

> **Note**
>
> 在运行此代码之前，请确保 quick-start 分类账中还没有名为 People 的活动表。
> 在第一行，将 *project-package* 替换为 步骤 1：设置您的 项目 中用于 Maven 命令的 groupId 值。

```
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
```

```
        });
    }
}
```

使用 Maven 或你的 IDE 来编译和运行该 App.java 文件。

Java 的 Amazon QLDB 驱动程序 — 说明书参考

本参考指南展示了 Java 的 Amazon QLDB 驱动程序的常见用例。它提供了 Java 代码示例，演示了如何使用该驱动程序运行基本的创建、读取、更新和删除 (CRUD) 操作。它还包括用于处理 Amazon Ion 数据的代码示例。此外，本指南还重点介绍了使事务具有幂等性和实现唯一性约束的最佳实践。

> **Note**
>
> 在适用的情况下，某些用例对于 Java 的 QLDB 驱动程序的每个支持主要版本都配备不同代码示例。

目录

- 创建 Ion 对象
- 读取 Ion 对象

导入驱动程序

以下代码示例导入驱动程序、QLDB 会话客户端、Amazon Ion 软件包以及其他相关依赖项。

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;
```

实例化驱动程序

以下代码示例创建了连接到指定分类账名称的驱动程序实例，并使用具有自定义重试限制的指定重试逻辑。

> **Note**
>
> 此示例还实例化了 Amazon Ion 系统对象 (`IonSystem`)。在本参考中运行某些数据操作时，您需要此对象来处理 Ion 数据。要了解更多信息，请参阅 使用 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();
```

CRUD 操作

QLDB 作为事务的一部分运行创建、读取、更新和删除 (CRUD) 操作。

> **Warning**
>
> 作为最佳实践，使写事务严格地幂等。

使事务幂等

我们建议将写事务设置为幂等，以避免重试时出现任何意想不到的副作用。如果事务可以运行多次并每次都产生相同的结果，则事务是幂等的。

例如，假设有一个事务，要将文档插入名为 Person 的表中。事务应该首先检查文档是否已经存在于表中。如果没有这种检查，表最终可能会有重复的文档。

假设 QLDB 在服务器端成功提交了事务，但客户端在等待响应时超时。如果事务不是幂等的，则在重试的情况下可以多次插入相同的文档。

使用索引避免全表扫描

我们还建议您在索引字段或文档 ID 上使用相等运算符来运行带有`WHERE`谓词子句的语句；例如，`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`。如果没有这种索引查找，QLDB 需要进行表扫描，这可能会导致事务超时或乐观并发控制 (OCC) 冲突。

有关 OCC 的更多信息，请参阅Amazon QLDB 并发模型。

隐式创建的事务

QldbDriver.execute 方法接受一个 lambda 函数，该函数接收一个 Executor 的实例，您可以用它来运行语句。`Executor`实例封装了隐式创建的事务。

您可以使用`Executor.execute`方法在 lambda 函数中运行语句。当 lambda 函数返回时，驱动程序会隐式提交事务。

以下各节介绍如何运行基本的 CRUD 操作、指定自定义重试逻辑以及如何实现唯一性约束。

> **Note**
>
> 在适用的情况下，这些部分提供了使用内置 Ion 库和 Jackson Ion 映射器库处理 Amazon Ion 数据代码示例。要了解更多信息，请参阅 使用 Amazon Ion。

目录

- 重试逻辑
- 实现唯一限制

创建表

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

创建索引

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

阅读文档

```
// 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
});
```

使用查询参数

以下代码示例使用 Ion 类型查询参数。

```
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
});
```

以下代码示例使用了多个查询参数。

```
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
});
```

以下代码示例使用查询参数列表。

```
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
});
```

使用 Jackson 映射程序

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

使用查询参数

以下代码示例使用 Ion 类型查询参数。

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

以下代码示例使用了多个查询参数。

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

以下代码示例使用查询参数列表。

```
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**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。在此示例中，我们建议在GovId字段上设置 索引以优化性能。如果不开启GovId索引，查询可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

插入文档

以下代码示例插入 Ion 数据类型。

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

使用 Jackson 映射程序

以下代码示例插入 Ion 数据类型。

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

此事务将文档插入 Person 表中。在插入之前，它首先检查文档是否已存在于表格内。此检查使事务本质上是幂等。即使您多次运行此事务，也不会造成任何异常副作用。

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一条语句内插入多个文档

要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个 IonList 类型的参数（显示转换为 IonValue），如下所示。

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

传递 IonList 列表时，不要将变量占位符 (?) 包括在双尖括号（<<...>>）内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

为什么需要显式转换？

TransactionExecutor.execute方法过载。它接受可变数量的 IonValue参数 (varargs) 或单个List<IonValue>参数。在 ion-java 中，IonList 被实现为List<IonValue>。

当您调用重载方法，Java 默认其为最具体的实施方法。在这种情况下，当您传递 IonList 参数时，它默认为采用List<IonValue>。调用时，此方法实现会将列表中的IonValue元素作为不同的值传递。因此，要改为调用 varargs 方法，必须将IonList参数显式转换为IonValue。

更新文档

```
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);
});
```

使用 Jackson 映射程序

```
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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

删除文档

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

使用 Jackson 映射程序

```
qldbDriver.execute(txn -> {
    try {
        txn.execute("DELETE FROM Person WHERE GovId = ?",
```

```
                MAPPER.writeValueAsIonValue("TOYENC486FH"));
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一个事务中运行多条语句

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

重试逻辑

驱动程序 execute 方法具有内置的重试机制，如果发生可重试的异常(例如超时或 OCC 冲突)，该机制可以重试事务。

2.x

最大重试次数和退避策略为可配置。

默认重试限值为4，默认退避策略为DefaultQldbTransactionBackoffStrategy。您可以使用 RetryPolicy 的实例为每个驱动程序实例以及每个事务设置重试策略。

以下代码示例使用自定义重试限制和驱动程序实例的自定义退避策略指定重试逻辑。

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

以下代码示例使用自定义重试限制和自定义回退策略指定重试逻辑。此execute配置将覆盖为驱动程序实例设置的重试逻辑。

```
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

最大重试次数。您可以通过在初始化PooledQldbDriver 时设置 retryLimit 属性来配置重试限制。

默认重试限制为 4。

实现唯一限制

QLDB 不支持唯一索引，但您可在应用程序中实现此行为。

假设您要对 Person 表中的 GovId 字段实现唯一性约束。据此，可以编写执行以下操作的事务：

1. 断言该表中没有指定 GovId 的现有文档。
2. 如果断言通过，请插入文档。

如果一个竞争事务同时通过断言，则只有一个事务将成功提交。另一笔事务将失败，并显示 OCC 冲突异常。

以下代码示例显示了如何实现此唯一约束。

```
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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

使用 Amazon Ion

可通过多种方法在 QLDB 中处理 Amazon Ion 数据。您可根据需要使用 Ion 库 中的内置方法，灵活创建和修改文档。或者，您可以使用 FasterXML 的 Jackson Ion 数据格式模块，以将 Ion 文档映射至普通旧 Java 对象 (POJO) 模型。

以下各节提供了使用这两种技术处理 Ion 数据的代码示例。

目录

导入 Ion 软件包

将工件 ion-java 作为依赖项添加至您的 Java 项目。

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>
```

导入以下 Ion 软件包。

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

使用 Jackson 映射程序

将 jackson-dataformat-ion 作为依赖项添加至您的 Java 项目中。QLDB 需要 `2.10.0` 版本或以上版本。

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>
```

导入以下 Ion 软件包。

```
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;
```

Ion 初始化

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

使用 Jackson 映射程序

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

创建 Ion 对象

以下代码示例使用 IonStruct 接口及其内置方法创建 Ion 对象。

```
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
```

使用 Jackson 映射程序

假设您有名为 JSON 映射的模型类Person，如下所示。

```
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;
    }
}
```

以下代码示例根据IonStruct实例创建对象Person。

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

读取 Ion 对象

以下代码示例打印 `ionStruct` 实例的每个字段。

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

使用 Jackson 映射程序

以下代码示例读取一个 `IonStruct` 对象并将其映射到的实例`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
```

有关使用 Ion 的更多信息，请参阅 GitHub 上的 Amazon Ion 文档。有关在 QLDB 中使用 Ion 的更多代码示例，请参阅使用 Amazon QLDB 中的 Amazon Ion 数据类型。

适用于 .NET 的 Amazon QLDB 驱动程序

要处理分类账中的数据，您可以使用AWS提供的驱动程序从 Microsoft .NET应用程序连接到 Amazon QLDB。分类账定位于 .NET 标准 2.0。更具体地说，它支持 .NET Core (LTS) 2.1+ 和 .NE T Framework 4.5.2+。有关兼容性的信息，请参阅 Microsoft 文档网站上的 .NET 标准。

我们强烈建议使用 Ion 对象映射器来完全无需在 Amazon Ion 类型和原生 C# 类型之间进行手动转换。

以下主题介绍了如何开始使用适用于 .NET 的 QLDB 驱动程序。

主题

- 驱动程序资源
- 先决条件
- 安装
- 适用于.NET 的 Amazon QLDB 驱动程序 — 快速入门教程

- 适用于 .NET 的 Amazon QLDB 驱动程序 — 说明书参考

驱动程序资源

有关 .NET 驱动程序支持功能的更多信息，请参阅以下资源：

- API 参考
- 驱动程序源代码 (GitHub)
- 示例应用程序源代码 (GitHub)
- Amazon Ion 说明书
- Ion 对象映射器 (GitHub)

先决条件

开始使用适用于 .NET 的 QLDB 驱动程序之前，您必须执行以下操作:

1. 按照 访问 Amazon QLDB 中的 AWS 的设置说明进行操作。这包括以下这些：
    1. 注册AWS。
    2. 创建具有适当 QLDB 权限的用户。
    3. 授权以编程方式访问开发。
2. 从微软 .NET 下载网站下载并安装 .NET Core SDK 2.1 或更高版本。
3. （可选）安装您选择的集成式开发环境 (IDE)，例如 Visual Studio、Mac 版 Visual Studio 或 Visual Studio Code。你可以从微软 Visual Studio 网站下载这些文件。
4. 配置您的开发环境用于 AWS SDK for .NET：
    1. 设置您的 AWS 凭证。我们建议创建共享的凭证文件。

        有关说明，请参阅AWS SDK for .NET开发者指南中的使用凭证文件配置 AWS 证书。
    2. 设置您的默认 AWS 区域。要了解如何操作，请参阅AWS 区域选择。

        有关可用区域的完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

接下来，您可以设置基本的示例应用程序并运行简短的代码示例，也可以将驱动程序安装到现有的 .NET 项目中。

- 要在现有项目中安装 QLDB 驱动程序和 AWS SDK for .NET，请继续执行安装。
- 要设置项目并运行演示分类账上基本数据事务的简短代码示例，请参阅 快速入门教程。

安装

使用 NuGet 包管理器安装适用于 .NET 的 QLDB 驱动程序。我们建议使用 Visual Studio 或您选择的 IDE 向项目添加依赖关系。驱动程序包名称为 Amazon.QLDB.Driver。

例如，在 Visual Studio 中，在 “工具” 菜单上打开 NuGet Package Manager 控制台。然后在 PM> 提示符处，输入以下命令。

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

安装驱动程序还会安装其依赖项，包括 AWS SDK for .NET 和 Amazon Ion 软件包。

安装 Ion 对象映射器

适用于 .NET 的 QLDB 驱动程序 1.3.0 版引入了无需使用 Amazon Ion 即可接受和返回原生 C# 数据类型的支持。要使用此功能，请将以下软件包添加到您的项目中。

- Amazon.QLDB.Driver.Serialization - 一个可以将 Ion 值映射到 C# 普通旧 CLR 对象 (POCO) 的库，反之亦然。此 Ion 对象映射器可让您的应用程序直接与原生 C# 数据类型进行交互，而无需使用 Ion。有关如何使用此库的简短指南，请参阅 GitHub 存储库 `awslabs/amazon-qldb-driver-dotnet` 中的 SERIALIZATION.md 文件。

要安装此程序包，请输入以下命令：

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

有关如何在分类账上运行基本数据事务的简短代码示例，请参阅 说明书参考。

适用于.NET 的 Amazon QLDB 驱动程序 — 快速入门教程

在本教程中，您将学习如何使用适用于.NET 的 Amazon QLDB 驱动程序来设置简单应用程序。本指南包括安装驱动程序的步骤以及创建、读取、更新和删除 (CRUD) 的基本操作的简短代码示例。

主题

- 先决条件

先决条件

在开始之前，请务必执行以下操作：

1. 请为 .NET 驱动程序完成 先决条件（如果尚未执行此操作）。这包括注册 AWS、授予开发所需的编程访问权限以及安装 .NET Core SDK。
2. 创建一个名为 `quick-start` 的分类账。

   要了解如何创建分类账，请参阅控制台入门中的 Amazon QLDB 分类账的基本操作 或 第 1 步：创建新分类账。

步骤 1：设置您的项目

首先，您需要设置您的 .NET 项目。

1. 要创建和运行模板应用程序，请在终端（例如 bash、Power Shell 或*dotnet*命令提示符）上输入以下命令。

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

   这个模版将创建一个名为`Amazon.QLDB.QuickStartGuide`的文件夹。在该文件夹中，它会创建一个同名项目和一个名为 `Program.cs` 的文件。该程序包含显示输出的代码 `Hello World!`。
2. 使用 NuGet 包管理器安装适用于 .NET 的 QLDB 驱动程序。我们建议使用 Visual Studio 或您选择的 IDE 向项目添加依赖关系。驱动程序包名称为 Amazon.QLDB.Driver。
   - 例如，在 Visual Studio 中，在 “工具” 菜单上打开 NuGet Package Manager 控制台。然后在 `PM>` 提示符处，输入以下命令。

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

- 或者，您可以在终端上输入以下命令。

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

安装驱动程序还会安装其依赖项，包括AWS SDK for .NET和 Amazon Ion 库。

3. 安装驱动程序的序列化库。

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

4. 打开 `Program.cs` 文件。

然后，按以下步骤中逐步添加代码示例，尝试一些基本的 CRUD 操作。或者，您可以跳过分步教程，改为运行完整的应用程序。

> **Note**
>
> - 在同步和异步 API 之间进行选择 — 驱动程序提供同步和异步 API。对于能够在不阻塞的情况下处理多个请求的高需求应用程序，我们建议使用异步 API 来提高性能。该驱动程序提供了同步 API，为同步编写的现有代码库提供了额外的便利。
>
>   本教程包括同步和异步代码示例。有关API的更多信息，请参阅 API 文档中的 IQldbDriver 和 IAsyncQldbDriver 接口。
> - 处理 Amazon Ion 数据 – 本教程提供了默认使用 Ion 对象映射器 处理 Amazon Ion 数据的代码示例。QLDB 在 .NET 驱动程序的 1.3.0 版本中引入 Ion 对象映射器。在适用的情况下，本教程还提供使用标准 Ion 库 作为替代方案的代码示例。要了解更多信息，请参阅 使用 Amazon Ion。

第 2 步：初始化驱动程序

初始化连接到名为 `quick-start` 的分类账的驱动程序实例。将以下代码添加到您的 `Program.cs` 文件。

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();
        }
    }
}
```

使用 Ion 库

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();
        }
    }
}
```

第 3 步：创建表和索引

在本教程至步骤 6 的其余部分中，您需要将以下代码示例附加到前面的代码示例中。

在此步骤中，以下代码显示了如何运行 `CREATE TABLE` 和 `CREATE INDEX` 语句。它会创建一个名为 `Person` 的表，并为该表上的 `firstName` 字段创建索引。索引是优化查询性能和帮助限制乐观并发控制 (OCC) 冲突异常所必需的。

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)");
});
```

第 4 步：插入文档

以下代码示例显示如何运行 INSERT 语句。QLDB 支持 PartiQL 查询语言（兼容 SQL）和 Amazon Ion 数据格式（JSON 的超集）。

添加以下代码，在 Person 表格中插入文档。

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);
});
```

使用 Ion 库

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

要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个Ion 列表类型的参数，如下所示。

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

传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

第 5 步：查询文档

以下代码示例显示如何运行 SELECT 语句。

添加以下代码，用于从 Person 表格中查询文档。

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
}
```

使用 Ion 库

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);
}
```

此示例使用问号 (?) 作为变量占位符，将文档信息传递给语句。使用占位符时，必须传递一个 IonValue 类型的值。

第 6 步：更新文档

以下代码示例显示如何运行 UPDATE 语句。

1. 添加以下代码，通过更新 age 到 42 来更新Person表格中的文档。

   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. 再次查询文档以查看更新的值。

   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. 要运行该应用程序，请在 Amazon.QLDB.QuickStartGuide 项目目录的父目录中输入以下命令。

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

使用 Ion 库

1. 添加以下代码，通过更新 age 到 42 来更新Person表格中的文档。

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. 再次查询文档以查看更新的值。

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. 要运行该应用程序，请在 Amazon.QLDB.QuickStartGuide 项目目录的父目录中输入以下命令。

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

运行完整的应用程序

以下代码示例是 Program.cs 应用程序的完整版本。您还可以从头到尾复制并运行此代码示例，而不必单独执行前面的步骤。此应用程序演示了对名为 quick-start 的分类账的一些基本的 CRUD 操作。

> **Note**
>
> 在运行此代码之前，请确保 quick-start 分类账中还没有名为 Person 的活动表。

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
            }

        }
    }
}
```

使用 Ion 库

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);
            }
        }
    }
}
```

要运行该应用程序，请在 Amazon.QLDB.QuickStartGuide 项目目录的父目录中输入以下命令。

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

适用于 .NET 的 Amazon QLDB 驱动程序 — 说明书参考

本参考指南显示了 .NET 的 Amazon QLDB 驱动程序的常见用例。它提供了 C# 代码示例，演示了如何使用该驱动程序运行基本的创建、读取、更新和删除 (CRUD) 操作。它还包括用于处理 Amazon Ion 数据的代码示例。此外，本指南还重点介绍了使事务具有幂等性和实现唯一性约束的最佳实践。

> **Note**
>
> 本主题提供了默认使用 Ion 对象映射器 处理 Amazon Ion 的代码示例。QLDB 在.NET 驱动程序的 1.3.0 版本中引入 Ion 对象映射器。在适用的情况下，本主题还提供了使用标准 Ion 库 作为替代方案代码示例。要了解更多信息，请参阅 使用 Amazon Ion。

目录

- 创建索引
- 阅读文档
  - 使用查询参数
- 插入文档
  - 在一条语句内插入多个文档
- 更新文档
- 删除文档
- 在一个事务中运行多条语句
- 重试逻辑
- 实现唯一限制
- 使用 Amazon Ion
  - 导入 Ion 模块
  - 创建 Ion 类型
  - 获取 Ion 二进制转储
  - 获取 Ion 文本转储

导入驱动程序

下面的代码示例导入驱动程序。

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

使用 Ion 库

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

实例化驱动程序

以下代码示例使用默认设置创建连接到指定分类账名称的驱动程序实例。

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();
```

使用 Ion 库

Async

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

Sync

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

CRUD 操作

QLDB 作为事务的一部分运行创建、读取、更新和删除 (CRUD) 操作。

> **Warning**
>
> 作为最佳实践，使写事务严格地幂等。

使事务幂等

我们建议将写事务设置为幂等，以避免重试时出现任何意想不到的副作用。如果事务可以运行多次并每次都产生相同的结果，则事务是幂等的。

例如，假设有一个事务，要将文档插入名为 Person 的表中。事务应该首先检查文档是否已经存在于表中。如果没有这种检查，表最终可能会有重复的文档。

假设 QLDB 在服务器端成功提交了事务，但客户端在等待响应时超时。如果事务不是幂等的，则在重试的情况下可以多次插入相同的文档。

使用索引避免全表扫描

我们还建议您在索引字段或文档 ID 上使用相等运算符来运行带有WHERE谓词子句的语句；例如，WHERE indexedField = 123或 WHERE indexedField IN (456, 789)。如果没有这种索引查找，QLDB 需要进行表扫描，这可能会导致事务超时或乐观并发控制 (OCC) 冲突。

有关 OCC 的更多信息，请参阅Amazon QLDB 并发模型。

隐式创建的事务

Amazon.QLDB.Driver.IQldbDriver.Execute 方法接受一个 lambda 函数，该函数接收一个 Amazon.QLDB.Driver.TransactionExecutor 的实例，您可以用它来运行语句。TransactionExecutor 的实例封装了隐式创建的事务。

您可以使用事务执行器的 Execute 执行方法在 lambda 函数中运行语句。当 lambda 函数返回时，驱动程序会隐式提交事务。

以下各节介绍如何运行基本的 CRUD 操作、指定自定义重试逻辑以及如何实现唯一性约束。

目录

- 实现唯一限制

创建表

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 }
}
```

使用 Ion 库

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"
    // }
}
```

创建索引

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 }
}
```

使用 Ion 库

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"
    // }
}
```

阅读文档

```
// 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**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。在此示例中，我们建议在GovId字段上设置 索引以优化性能。如果不开启GovId索引，查询可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

使用查询参数

以下代码示例使用 C# 类型查询参数。

```
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.
}
```

以下代码示例使用了多个 C# 类型查询参数。

```
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.
}
```

以下代码示例使用 C# 类型查询参数。

```
// 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.
}
```

以下代码示例使用 C# 列表为值。

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

使用 Ion 库

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**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。在此示例中，我们建议在GovId字段上设置 索引以优化性能。如果不开启GovId索引，查询可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

以下代码示例使用 Ion 类型查询参数。

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.
}
```

以下代码示例使用了多个查询参数。

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.
}
```

以下代码示例使用查询参数列表。

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.
}
```

以下代码示例使用 C# 列表为值。要了解使用不同 Ion 类型任务的更多信息，请参阅 使用 Amazon QLDB 中的 Amazon Ion 数据类型。

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"
    //     }
    //     ]
    // }
}
```

插入文档

以下代码示例插入 Ion 数据类型。

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

使用 Ion 库

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);
});
```

此事务将文档插入 Person 表中。在插入之前，它首先检查文档是否已存在于表格内。此检查使事务本质上是幂等。即使您多次运行此事务，也不会造成任何异常副作用。

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一条语句内插入多个文档

要使用单个 INSERT 语句插入多个文档，可以按如下方式向该语句传递 C# `List` 参数。

```
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 }
}
```

使用 Ion 库

要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个Ion 列表类型的参数，如下所示。

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"
    // }
}
```

传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

更新文档

```
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 }
}
```

使用 Ion 库

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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

删除文档

```
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 }
}
```

使用 Ion 库

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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一个事务中运行多条语句

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

使用 Ion 库

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;
    });
}
```

重试逻辑

有关驱动程序的内置重试逻辑的信息，请参见 使用 Amazon QLDB 中的驱动程序了解重试策略。

实现唯一限制

QLDB 不支持唯一索引，但您可在应用程序中实现此行为。

假设您要对 Person 表中的 GovId 字段实现唯一性约束。据此，可以编写执行以下操作的事务：

1. 断言该表中没有指定 GovId 的现有文档。
2. 如果断言通过，请插入文档。

如果一个竞争事务同时通过断言，则只有一个事务将成功提交。另一笔事务将失败，并显示 OCC 冲突异常。

以下代码示例显示了如何实现此唯一约束。

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

使用 Ion 库

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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

使用 Amazon Ion

可通过多种方法在 QLDB 中处理 Amazon Ion 数据。您可使用 Ion 库创建和修改 Ion 值。或者，您可以使用 Ion 对象映射器将 C# 普通旧 CLR 对象 (POCO) 映射进出 Ion 值。适用 .NET 的 Amazon QLDB 驱动程序 1.3 版本引入了对 Ion 对象映射器的支持。

以下各节提供了使用这两种技术处理 Ion 数据的代码示例。

目录

导入 Ion 模块

```
using Amazon.IonObjectMapper;
```

使用 Ion 库

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

创建 Ion 类型

以下代码示例介绍了如何使用 Ion 对象映射器从 C# 对象创建 Ion 值。

```
// 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);
```

使用 Ion 库

以下代码示例介绍了使用 Ion 库创建 Ion 值的两种方法。

使用 **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);
```

使用 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);
```

获取 Ion 二进制转储

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

使用 Ion 库

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

获取 Ion 文本转储

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

使用 Ion 库

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

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

有关使用 Ion 的更多信息，请参阅 GitHub 上的 Amazon Ion 文档。有关在 QLDB 中使用 Ion 的更多代码示例，请参阅使用 Amazon QLDB 中的 Amazon Ion 数据类型。

适用于 Go 的 Amazon QLDB 驱动程序

要处理分类账中的数据，您可以使用AWS提供的驱动程序从 Go 应用程序连接到 Amazon QLDB。以下主题介绍了如何开始使用适用于 Go 的 QLDB 驱动程序。

主题

驱动程序资源

有关 Go 驱动程序支持功能的更多信息，请参阅以下资源：

- API 参考：3.x、2.x、1.x
- 驱动程序源代码 (GitHub)
- Amazon Ion 说明书

先决条件

开始使用适用于 Go 的 QLDB 驱动程序之前，您必须执行以下操作:

1. 按照 访问 Amazon QLDB 中的 AWS 的设置说明进行操作。这包括以下这些：
    1. 注册AWS。
    2. 创建具有适当 QLDB 权限的用户。
    3. 授权以编程方式访问开发。
2. （可选）安装您选择的集成式开发环境 (IDE)。有关 Go 常用 IDE 的列表，请参阅 Go 网站上的编辑器插件和 IDE。
3. 从 Go 下载网站下载并安装以下 Go 版本之一：
    - 1.15 或更高版本 - 适用于 Go v3 的 QLDB 驱动程序
    - 1.14 - 适用于 Go v1 或 v2 的 QLDB 驱动程序

4. 配置您的开发环境用于 AWS SDK for Go：

   1. 设置您的 AWS 凭证。我们建议创建共享的凭证文件。

      有关说明，请参阅AWS SDK for Go开发者指南中的指定凭证。

   2. 设置您的默认 AWS 区域。要了解如何操作，请参阅指定AWS 区域。

      有关可用区域的完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

接下来，您可以设置基本的示例应用程序并运行简短的代码示例，也可以将驱动程序安装到现有的 Go 项目中。

- 要在现有项目中安装 QLDB 驱动程序和 AWS SDK for Go，请继续执行安装。
- 要设置项目并运行演示分类账上基本数据事务的简短代码示例，请参阅 快速入门教程。

安装

Go 的 QLDB 驱动程序在 GitHub 存储库 awslabs/amazon-qldb-driver-go 中是开源的。QLDB 支持以下驱动程序版本及其 Go 依赖项。

驱动程序版本	Go 版本	状态	最新发布日期
1.x	1.14 或更高版本	量产版	2021 年 6 月 16 日
2.x	1.14 或更高版本	量产版	2021 年 7 月 21 日
3.x	1.15 或更高版本	量产版	2022 年 11 月 10 日

安装驱动程序

1. 确保您的项目使用 Go 模块来安装项目依赖项。
2. 在您的项目目录中输入以下 `go get` 命令。

   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
```

安装驱动程序还会安装其依赖项，包括 AWS SDK for Go 或 AWS SDK for Go v2 和 Amazon Ion软件包。

有关如何在分类账上运行基本数据事务的简短代码示例，请参阅 说明书参考。

适用于 Go 的 Amazon QLDB 驱动程序 — 快速入门教程

在本教程中，您将学习如何使用适用于 Go 的最新版 Amazon QLDB 驱动程序来设置简单应用程序。本指南包括安装驱动程序的步骤以及创建、读取、更新和删除 (CRUD) 的基本操作的简短代码示例。

主题

先决条件

在开始之前，请务必执行以下操作：

1. 请为 Go 驱动程序完成 先决条件（如果尚未执行此操作）。这包括注册 AWS、授予开发所需的编程访问权限以及安装 Go。
2. 创建一个名为 `quick-start` 的分类账。

要了解如何创建分类账，请参阅控制台入门中的 Amazon QLDB 分类账的基本操作 或 第 1 步：创建新分类账。

步骤 1：安装驱动程序

确保您的项目使用 Go 模块来安装项目依赖项。

在您的项目目录中运行以下 go get 命令。

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

安装驱动程序还会安装其依赖项，包括 AWS SDK for Go v2 和 Amazon Ion 软件包。

第 2 步：导入软件包

导入以下 AWS 软件包。

```
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"
)
```

第 3 步：初始化驱动程序

初始化连接到名为 quick-start 的分类账的驱动程序实例。

```
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**
>
> 在此代码示例中，将 *us-east-1* 替换为 AWS 区域（即您创建分类账的位置）。

第 4 步：创建表和索引

以下代码示例显示如何运行 CREATE TABLE 和 CREATE INDEX 语句。

```
_, 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)
}
```

它会创建一个名为 People 的表，并为该表上的 firstName 和 age 字段创建索引。索引是优化查询性能和帮助限制乐观并发控制 (OCC) 冲突异常所必需的。

第 5 步：插入文档

以下代码示例显示如何运行 INSERT 语句。QLDB 支持 PartiQL 查询语言（兼容 SQL）和 Amazon Ion 数据格式（JSON 的超集）。

使用文字 PartiQL

以下代码使用字符串文字 PartiQL 语句将文档插入 People 表中。

```
_, 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)
}
```

使用 Ion 数据类型

与 Go 内置的 JSON 包类似，您可以在 Ion 中封送和解封送 Go 数据类型。

1. 假设您具有以下名为 Person 的 Go 结构。

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

2. 创建 Person 的实例。

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

   驱动程序会为您编组 person 的一个 ION 编码的文本表示形式。

> **Important**
>
> 要使 marshal 和 unmarshal 正常工作，必须导出 Go 数据结构的字段名称（首字母大写）。

3. 将 `person` 实例传递给事务`Execute`的方法。

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

此示例使用问号 (?) 作为变量占位符，将文档信息传递给语句。使用占位符时，必须传递 ION 编码的文本值。

> **Tip**
>
> 要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个列表类型的参数，如下所示。
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> 传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

第 6 步：查询文档

以下代码示例显示如何运行 SELECT 语句。

```
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")
}
```

此示例从 People 表中查询您的文档，假设结果集不为空，然后从结果中返回您的文档。

步骤 7：更新文档

以下代码示例显示如何运行 UPDATE 语句。

```
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)
}
```

步骤 8：查询更新的文档

以下代码示例通过 `firstName` 查询 `People` 表，并返回结果集中的所有文档。

```
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")
}
```

第 9 步：删除表格

以下代码示例显示如何运行 DROP TABLE 语句。

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

运行完整的应用程序

以下代码示例是应用程序的完整版本。您还可以从头到尾复制并运行此代码示例，而不必单独执行前面的步骤。此应用程序演示了对名为 quick-start 的分类账的一些基本的 CRUD 操作。

> **Note**
>
> 在运行此代码之前，请确保 quick-start 分类账中还没有名为 People 的活动表。

```
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)
    }
}
```

适用于 Go 的 Amazon QLDB 驱动程序 — 说明书参考

本参考指南显示了 Go 的 Amazon QLDB 驱动程序的常见用例。它提供了 Go 代码示例，演示了如何使用该驱动程序运行基本的创建、读取、更新和删除 (CRUD) 操作。它还包括用于处理 Amazon Ion 数据的代码示例。此外，本指南还重点介绍了使事务具有幂等性和实现唯一性约束的最佳实践。

> **Note**
>
> 在适用的情况下，某些用例对于 Go 的 QLDB 驱动程序的每个支持主要版本都配备不同代码示例。

目录

- 创建表
- 创建索引
- 阅读文档
  - 使用查询参数
- 插入文档
  - 在一条语句内插入多个文档
- 更新文档
- 删除文档
- 在一个事务中运行多条语句
- 重试逻辑
- 实现唯一限制
- 使用 Amazon Ion
  - 导入 Ion 模块
  - 创建 Ion 类型
  - 获取二进制 Ion
  - 获取 Ion 文本

导入驱动程序

以下代码示例导入驱动程序和其他必需 AWS 软件包。

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**
>
> 此示例还导入了 Amazon Ion 软件包 (amzn/ion-go/ion)。在本参考中运行某些数据操作时，您需要此软件包来处理 Ion 数据。要了解更多信息，请参阅 使用 Amazon Ion。

实例化驱动程序

以下代码示例在AWS 区域中创建连接到指定分类账名称的驱动程序实例。

3.x

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic(err)
}

qldbSession := qldbsession.NewFromConfig(cfg, func(options *qldbsession.Options) {
    options.Region = "us-east-1"
})
driver, err := qldbdriver.New(
  "vehicle-registration",
  qldbSession,
  func(options *qldbdriver.DriverOptions) {
    options.LoggerVerbosity = qldbdriver.LogInfo
})
if err != nil {
  panic(err)
}

defer driver.Shutdown(context.Background())
```

2.x

```
awsSession := session.Must(session.NewSession(aws.NewConfig().WithRegion("us-
east-1")))
qldbSession := qldbsession.New(awsSession)

driver, err := qldbdriver.New(
  "vehicle-registration",
  qldbSession,
  func(options *qldbdriver.DriverOptions) {
    options.LoggerVerbosity = qldbdriver.LogInfo
  })
if err != nil {
  panic(err)
}
```

CRUD 操作

QLDB 作为事务的一部分运行创建、读取、更新和删除 (CRUD) 操作。

> **Warning**
>
> 作为最佳实践，使写事务严格地幂等。

使事务幂等

我们建议将写事务设置为幂等，以避免重试时出现任何意想不到的副作用。如果事务可以运行多次并每次都产生相同的结果，则事务是幂等的。

例如，假设有一个事务，要将文档插入名为 Person 的表中。事务应该首先检查文档是否已经存在于表中。如果没有这种检查，表最终可能会有重复的文档。

假设 QLDB 在服务器端成功提交了事务，但客户端在等待响应时超时。如果事务不是幂等的，则在重试的情况下可以多次插入相同的文档。

使用索引避免全表扫描

我们还建议您在索引字段或文档 ID 上使用相等运算符来运行带有WHERE谓词子句的语句；例如，`WHERE indexedField = 123`或 `WHERE indexedField IN (456, 789)`。如果没有这种索引查找，QLDB 需要进行表扫描，这可能会导致事务超时或乐观并发控制 (OCC) 冲突。

有关 OCC 的更多信息，请参阅Amazon QLDB 并发模型。

隐式创建的事务

QLDBDriver.Execute 函数接受一个 lambda 函数，该函数接收一个 Transaction 的实例，您可以用它来运行语句。的实例Transaction封装了隐式创建的事务。

您可以使用Transaction.Execute函数在 lambda 函数中运行语句。当 lambda 函数返回时，驱动程序会隐式提交事务。

以下各节介绍如何运行基本的 CRUD 操作、指定自定义重试逻辑以及如何实现唯一性约束。

目录

创建表

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE TABLE Person")
})
```

创建索引

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("CREATE INDEX ON Person(GovId)")
```

```
})
```

阅读文档

```
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)
}
```

使用查询参数

以下代码示例使用原生类型查询参数。

```
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)
}
```

以下代码示例使用了多个查询参数。

```
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)
}
```

以下代码示例使用查询参数列表。

```
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**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。在此示例中，我们建议在GovId字段上设置 索引以优化性能。如果不开启GovId索引，查询可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

插入文档

以下代码示例插入本地数据类型。

```
_, 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
})
```

此事务将文档插入 Person 表中。在插入之前，它首先检查文档是否已存在于表格内。此检查使事务本质上是幂等。即使您多次运行此事务，也不会造成任何异常副作用。

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一条语句内插入多个文档

要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个列表类型的参数，如下所示。

```
// people is a list
txn.Execute("INSERT INTO People ?", people)
```

传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

更新文档

以下代码示例使用原生数据类型。

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
```

```
  return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?", "John",
 "TOYENC486FH")
})
```

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

删除文档

以下代码示例使用原生数据类型。

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("DELETE FROM Person WHERE GovId = ?", "TOYENC486FH")
})
```

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一个事务中运行多条语句

```
// 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
}
```

重试逻辑

驱动程序 `Execute` 函数具有内置的重试机制，如果发生可重试的异常(例如超时或 OCC 冲突)，该机制可以重试事务。最大重试次数和退避策略为可配置。

默认的重试限制为 4，默认的退避策略是以10毫秒为基数的ExponentialBackoffStrategy。您可以使用 retryPolicy 实例为每个驱动程序实例以及每个事务设置重试策略。

以下代码示例使用自定义重试限制和驱动程序实例的自定义退避策略指定重试逻辑。

```
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)
  }
}
```

以下代码示例使用自定义重试限制和自定义回退策略指定重试逻辑，用于特定匿名函数。该 SetRetryPolicy 函数将覆盖为驱动程序实例设置的重试策略。

```
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")
  })
}
```

实现唯一限制

QLDB 不支持唯一索引，但您可在应用程序中实现此行为。

假设您要对 Person 表中的 GovId 字段实现唯一性约束。据此，可以编写执行以下操作的事务：

1. 断言该表中没有指定 GovId 的现有文档。
2. 如果断言通过，请插入文档。

如果一个竞争事务同时通过断言，则只有一个事务将成功提交。另一笔事务将失败，并显示 OCC 冲突异常。

以下代码示例显示了如何实现此唯一约束。

```
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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

使用 Amazon Ion

以下各节说明了如何使用 Amazon Ion 模块处理 Ion 数据。

目录

导入 Ion 模块

```
import "github.com/amzn/ion-go/ion"
```

创建 Ion 类型

Go 版 Ion 库当前不支持文档对象模型 (DOM)，因此您无法创建 Ion 数据类型。但是使用 QLDB 时，您可以在 Go 原生类型和 Ion 二进制文件之间进行编组和解组。

获取二进制 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]
```

获取 Ion 文本

```
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"}
```

有关 Ion 更多信息，请参阅 GitHub 上的Amazon Ion 文档。有关在 QLDB 中使用 Ion 的更多代码示例，请参阅使用 Amazon QLDB 中的 Amazon Ion 数据类型。

适用于 Node.js 的 Amazon QLDB 驱动程序

要处理分类账中的数据，您可以使用AWS提供的驱动程序从 Node.js 应用程序连接到 Amazon QLDB。以下主题介绍了如何开始使用 Node.js 上 QLDB 驱动程序。

主题

驱动程序资源

有关 Node.js 驱动程序支持功能的更多信息，请参阅以下资源：

- API 参考：3.x、2.x、1.x
- 驱动程序源代码 (GitHub)
- 示例应用程序源代码 (GitHub)
- Amazon Ion 代码示例
- 使用 AWS Lambda在 QLDB 上构建简单 CRUD 操作和数据流（AWS 博客）

先决条件

开始使用适用于 Node.js 的 QLDB 驱动程序之前，您必须执行以下操作:

1. 按照 访问 Amazon QLDB 中的 AWS 的设置说明进行操作。这包括以下这些：
    1. 注册AWS。
    2. 创建具有适当 QLDB 权限的用户。
    3. 授权以编程方式访问开发。
2. 从 Node.js 下载 网站安装 Node.js 14.x 或更高版本。（该驱动程序先前版本支持 Node.js 版本 10.x 或更高版本。）

3. 为AWSNode.js 中的 JavaScript SDK配置开发环境：

   1. 设置您的 AWS 凭证。我们建议创建共享的凭证文件。

      有关说明，请参阅AWS SDK for JavaScript 开发指南中的从共享凭据文件加载 Node.js 中的凭据。

   2. 设置您的默认 AWS 区域。要了解如何使用，请参阅 设置AWS 区域。

      有关可用区域的完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

接下来，您可下载完整的教程示例应用程序，也可以只在 Node.js 项目中安装驱动程序并运行短代码示例。

- 要在现有项目 Node.js 中安装 QLDB 驱动程序和适用于 JavaScript 的 AWSSDK，请继续执行。安装
- 要设置项目并运行演示分类账上基本数据事务的简短代码示例，请参阅 快速入门教程。
- 要在完整的教程示例应用程序中运行更深入的数据和管理 API 操作示例，请参阅 Node.js 教程。

安装

QLDB 支持以下驱动程序版本及其 Node.js 依赖项。

驱动程序版本	Node.js 版本	状态	最新发布日期
1.x	10.x 或更高版本	量产版	2020 年 6 月 5 日
2.x	10.x 或更高版本	量产版	2021 年 5 月 6 日
3.x	14.x 或更高版本	量产版	2023 年 11 月 10 日

要使用 npm (Node.js 包管理器)安装 QLDB 驱动程序，请从项目根目录中输入以下命令。

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
```

驱动程序对以下软件包具有对等依赖项。您还必须将这些软件包作为依赖项安装至项目中。

3.x

模块化聚合 QLDB 客户端 (管理 API)

```
npm install @aws-sdk/client-qldb
```

模块化聚合 QLDB 会话客户端 (数据 API)

```
npm install @aws-sdk/client-qldb-session
```

Amazon Ion 数据格式

```
npm install ion-js
```

`BigInt`纯 JavaScript 实施

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Amazon Ion 数据格式

```
npm install ion-js@4.0.0
```

`BigInt`纯 JavaScript 实施

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Amazon Ion 数据格式

```
npm install ion-js@4.0.0
```

`BigInt`纯 JavaScript 实施

```
npm install jsbi@3.1.1
```

使用驱动程序连接至分类账

然后，您可以导入驱动程序，并使用它来连接到分类账。以下 TypeScript 代码示例说明如何为指定的分类账名称和AWS 区域创建驱动程序实例。

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);
});
```

有关如何在分类账上运行基本数据交易的简短代码示例，请参阅 说明书参考。

设置建议

重复使用具有保持连接功能的连接

QLDB Node.js 驱动程序 v3

默认的 Node.js HTTP/HTTPS 代理会为每个新请求创建一个新的 TCP 连接。为了避免重建连接的成本，AWS SDK for JavaScriptv3 默认会重复使用 TCP 连接。有关更多信息以及如何禁用 “重用” 连接，请参阅AWS SDK for JavaScript 开发人员指南中的在 Node.js 中通过 keep-alive 重用连接。

我们建议使用默认设置来重用 Node.js 的 QLDB 驱动程序中的连接。在驱动程序初始化期间，将低级客户端 HTTP 选项 `maxSockets` 设置为`maxConcurrentTransactions` 与您设置的值相同。

例如，请参见以下 JavaScript 或 TypeScript 代码。

JavaScript

```
const qldb = require('amazon-qldb-driver-nodejs');
const https = require('https');

//Replace this value as appropriate for your application
const maxConcurrentTransactions = 50;

const agentForQldb = new https.Agent({
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    "maxSockets": maxConcurrentTransactions
});

const lowLevelClientHttpOptions = {
    httpAgent: agentForQldb
}

let driver = new qldb.QldbDriver("testLedger", undefined, lowLevelClientHttpOptions,
 maxConcurrentTransactions);
```

TypeScript

```
import { Agent } from 'https';
import { NodeHttpHandlerOptions } from "@aws-sdk/node-http-handler";
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

//Replace this value as appropriate for your application
const maxConcurrentTransactions: number = 50;

const agentForQldb: Agent = new Agent({
    //Set this to the same value as `maxConcurrentTransactions`(previously called
 `poolLimit`)
    //Do not rely on the default value of `Infinity`
    maxSockets: maxConcurrentTransactions
});

const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: agentForQldb
};
```

```
let driver = new QldbDriver("testLedger", undefined, lowLevelClientHttpOptions,
 maxConcurrentTransactions);
```

QLDB Node.js 驱动程序 v2

默认的 Node.js HTTP/HTTPS 代理会为每个新请求创建一个新的 TCP 连接。为了节省建立新连接的成本，建议重复使用现有的连接。

要重用 Node.js 的 QLDB 驱动程序中的连接，请使用以下选项之一：

- 在驱动程序初始化时，设置以下低级客户端 HTTP 选项：
  - `keepAlive` – `true`
  - `maxSockets` — 与您设置的`maxConcurrentTransactions`值相同

  例如，请参见以下 JavaScript 或 TypeScript 代码。

  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);
```

- 或者，您也可以设置 AWS_NODEJS_CONNECTION_REUSE_ENABLED 环境变量为 1。有关示例，请参阅 AWS SDK for JavaScript 开发人员指南 中的在 Node.js 中重复使用具有保持连接功能的连接。

> **Note**
>
> 如果您设置此环境变量，它将影响所有使用AWS SDK for JavaScript的AWS 服务。

适用于 Node.js 的 Amazon QLDB 驱动程序 — 快速入门教程

在本教程中，您将学习如何使用适用于 Node.js 的 Amazon QLDB 驱动程序来设置简单应用程序。本指南包括安装驱动程序的步骤以及创建、读取、更新和删除 (CRUD) 的基本操作的简短 JavaScript 和 TypeScript 代码示例。有关在完整示例应用程序中演示这些操作的更深入的示例，请参阅 Node.js 教程。

> **Note**
>
> 在适用的情况下，某些步骤对于 Node.js 的 QLDB 驱动程序的每个支持主要版本都配备不同代码示例。

主题

先决条件

在开始之前，请务必执行以下操作：

1. 请为 Node.js 驱动程序完成 先决条件（如果尚未执行此操作）。这包括注册 AWS、授予开发所需的编程访问权限以及安装 Node.js。
2. 创建一个名为 `quick-start` 分类账。

   要了解如何创建分类账，请参阅控制台入门中的 Amazon QLDB 分类账的基本操作 或 第 1 步：创建新分类账。

如果您使用的是 TypeScript，则还必须执行以下设置步骤。

使用 TypeScript

安装 TypeScript

1. 安装 TypeScript 包。QLDB 驱动程序在 TypeScript 3.8.x 上运行。

```
$ npm install --global typescript@3.8.0
```

2. 软件包安装完成后，请运行以下命令以确保TypeScript编译器已安装。

```
$ tsc --version
```

要按照以下步骤运行代码，请注意必须先将 TypeScript 文件转换为可执行的 JavaScript 代码，如下所示。

```
$ tsc app.ts; node app.js
```

步骤 1：设置您的项目

首先，您需要设置您的 Node.js 项目。

1. 为应用程序创建一个文件夹。

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

2. 要初始化您的项目，请输入以下 npm 命令并回答安装过程中提出的问题。您可以对大多数问题使用默认值。

```
$ npm init
```

3. 安装适用于 Node.js 的Amazon QLDB 驱动程序。
   - 使用版本 3.x

```
$ npm install amazon-qldb-driver-nodejs --save
```

   - 使用版本 2.x

```
$ npm install amazon-qldb-driver-nodejs@2.2.0 --save
```

   - 使用版本 1.x

```
$ npm install amazon-qldb-driver-nodejs@1.0.0 --save
```

4. 安装驱动程序的对等依赖项。
   - 使用版本 3.x

```
$ npm install @aws-sdk/client-qldb-session --save
$ npm install ion-js --save
$ npm install jsbi --save
```

- 使用版本 2.x 或 1.x

```
$ npm install aws-sdk --save
$ npm install ion-js@4.0.0 --save
$ npm install jsbi@3.1.1 --save
```

5. 创建一个名为 `app.js`（适用于 JavaScript）或 `app.ts`（适用于 TypeScript）的新文件。

   然后，按以下步骤中逐步添加代码示例，尝试一些基本的 CRUD 操作。或者，您可以跳过分步教程，改为运行完整的应用程序。

第 2 步：初始化驱动程序

初始化连接到名为 `quick-start` 的分类账的驱动程序实例。将以下代码添加到您的 `app.js` 或 `app.ts` 文件。

使用版本 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**
>
> - 在此代码示例中，将 *us-east-1* 替换为 AWS 区域（即您创建分类账的位置）。
> - 为简单起见，本指南中的其余代码示例使用具有默认设置的驱动程序，如以下版本 1.x 示例中所述。您也可以使用您自己的驱动程序实例来代替自定义 RetryConfig。

使用版本 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

- 在此代码示例中，将 *us-east-1* 替换为 AWS 区域（即您创建分类账的位置）。
- 版本 2.x 引入了用于初始化 `QldbDriver` 的新可选参数 `RetryConfig`。
- 为简单起见，本指南中的其余代码示例使用具有默认设置的驱动程序，如以下版本 1.x 示例中所述。您也可以使用您自己的驱动程序实例来代替自定义 `RetryConfig`。
- 此代码示例初始化一个驱动程序，该驱动程序通过设置 keep-alive 选项来重用现有连接。要了解更多信息，请参阅 设置建议 了解 Node.js 驱动程序。

使用版本 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**
>
> 您可以设置 AWS_REGION 环境变量以确定区域。有关更多信息，请参阅《AWS SDK for JavaScript 开发人员指南》中的 设置AWS 区域。

第 3 步：创建表和索引

以下代码示例显示如何运行 CREATE TABLE 和 CREATE INDEX 语句。

1. 添加以下函数，创建名为 People 的表。

   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. 添加以下函数，为 People 表中的 firstName 字段创建索引。索引是优化查询性能和帮助限制乐观并发控制 (OCC) 冲突异常所必需的。

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. 在 main 函数中，你先调用 createTable，然后调用 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. 运行代码以创建表和索引。

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

第 4 步：插入文档

以下代码示例显示如何运行 INSERT 语句。QLDB 支持 PartiQL 查询语言（兼容 SQL）和 Amazon Ion 数据格式（JSON 的超集）。

1. 添加以下函数，在 People 表格中插入文档。

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);
}
```

此示例使用问号 (?) 作为变量占位符，将文档信息传递给语句。该 execute 方法同时支持 Amazon Ion 类型和 Node.js 本机类型。

> **Tip**
>
> 要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个列表类型的参数，如下所示。
>
> ```
> // people is a list
> txn.execute("INSERT INTO People ?", people);
> ```
>
> 传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

2. 在 `main` 函数中，移除 `createTable` 和 `createIndex` 调用，然后向添加调用 `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();
}
```

第 5 步：查询文档

以下代码示例显示如何运行 SELECT 语句。

1. 添加以下函数，用于从 People 表格中查询文档。

   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. 在 main 函数中，在对的调用 fetchDocuments 之后添加以下调用 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();
}
```

第 6 步：更新文档

以下代码示例显示如何运行 UPDATE 语句。

1. 添加以下函数，通过修改 `lastName` 到 `"Stiles"` 来更新 `People` 表中的文档。

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. 在 main 函数中，在对的调用 updateDocuments 之后添加以下调用 fetchDocuments。然后，再次调用 fetchDocuments 来查看更新的结果。

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. 运行代码以插入、查询和更新文档。

   JavaScript

   ```
   $ node app.js
   ```

   TypeScript

   ```
   $ tsc app.ts; node app.js
   ```

运行完整的应用程序

以下代码示例是完整版本的 `app.js` 和 `app.ts`。您还可以从头到尾运行此代码，而不必单独执行前面的步骤。此应用程序演示了对名为 `quick-start` 的分类账的一些基本的 CRUD 操作。

> **Note**
>
> 在运行此代码之前，请确保 quick-start 分类账中还没有名为 People 的活动表。

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();
}
```

要运行完整的应用程序，请输入以下命令。

JavaScript

```
$ node app.js
```

TypeScript

```
$ tsc app.ts; node app.js
```

Node.js 的 Amazon QLDB 驱动程序 — 说明书参考

本参考指南显示了 Node.js 的 Amazon QLDB 驱动程序的常见用例。它提供了 JavaScript 和 TypeScript 代码示例，演示了如何使用该驱动程序运行基本的创建、读取、更新和删除 (CRUD) 操作。它还包括用于处理 Amazon Ion 数据的代码示例。此外，本指南还重点介绍了使事务具有幂等性和实现唯一性约束的最佳实践。

目录

- 获取 Ion 二进制转储
- 获取 Ion 文本转储

导入驱动程序

下面的代码示例导入驱动程序。

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**
>
> 此示例还导入了 Amazon Ion 软件包 (`ion-js`)。在本参考中运行某些数据操作时，您需要此软件包来处理 Ion 数据。要了解更多信息，请参阅 使用 Amazon Ion。

实例化驱动程序

以下代码示例使用默认设置创建连接到指定分类账名称的驱动程序实例。

JavaScript

```
const qldbDriver = new qldb.QldbDriver("vehicle-registration");
```

TypeScript

```
const qldbDriver: QldbDriver = new QldbDriver("vehicle-registration");
```

CRUD 操作

QLDB 作为事务的一部分运行创建、读取、更新和删除 (CRUD) 操作。

> **Warning**
>
> 作为最佳实践，使写事务严格地幂等。

使事务幂等

我们建议将写事务设置为幂等，以避免重试时出现任何意想不到的副作用。如果事务可以运行多次并每次都产生相同的结果，则事务是幂等的。

例如，假设有一个事务，要将文档插入名为 `Person` 的表中。事务应该首先检查文档是否已经存在于表中。如果没有这种检查，表最终可能会有重复的文档。

假设 QLDB 在服务器端成功提交了事务，但客户端在等待响应时超时。如果事务不是幂等的，则在重试的情况下可以多次插入相同的文档。

使用索引避免全表扫描

我们还建议您在索引字段或文档 ID 上使用相等运算符来运行带有`WHERE`谓词子句的语句；例如，`WHERE indexedField = 123`或 `WHERE indexedField IN (456, 789)`。如果没有这种索引查找，QLDB 需要进行表扫描，这可能会导致事务超时或乐观并发控制 (OCC) 冲突。

有关 OCC 的更多信息，请参阅Amazon QLDB 并发模型。

隐式创建的事务

QldbDriver.executeLambda 方法接受一个 lambda 函数，该函数接收一个 TransactionExecutor 的实例，您可以用它来运行语句。`TransactionExecutor` 的实例封装了隐式创建的事务。

您可以使用事务执行器的执行方法在 lambda 函数中运行语句。当 lambda 函数返回时，驱动程序会隐式提交事务。

> **Note**
>
> 该 `execute` 方法同时支持 Amazon Ion 类型和 Node.js 本机类型。如果您将 Node.js 原生类型作为参数传递给 `execute`，则驱动程序会使用该 `ion-js` 包将其转换为 Ion 类型（前

提是支持对给定 Node.js 数据类型的转换）。有关支持的数据类型和转换规则，请参阅 Ion JavaScript DOM README。

以下各节介绍如何运行基本的 CRUD 操作、指定自定义重试逻辑以及如何实现唯一性约束。

目录

创建表

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

创建索引

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

阅读文档

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']
        }
    });
}());
```

使用查询参数

以下代码示例使用原生类型查询参数。

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']
        }
    });
}());
```

以下代码示例使用 Ion 类型查询参数。

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']
        }
    });
}());
```

以下代码示例使用了多个查询参数。

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']
        }
    });
}());
```

以下代码示例使用查询参数列表。

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**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。在此示例中，我们建议在GovId字段上设置 索引以优化性能。如果不开启GovId索引，查询可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

插入文档

以下代码示例插入本地数据类型。

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

以下代码示例插入 Ion 数据类型。

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);
        }
```

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

此事务将文档插入 Person 表中。在插入之前，它首先检查文档是否已存在于表格内。此检查使事务本质上是幂等。即使您多次运行此事务，也不会造成任何异常副作用。

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一条语句内插入多个文档

要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个列表类型的参数，如下所示。

```
// people is a list
txn.execute("INSERT INTO People ?", people);
```

传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

更新文档

以下代码示例使用原生数据类型。

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');
```

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

以下代码示例使用 Ion 数据类型。

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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

删除文档

以下代码示例使用原生数据类型。

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

以下代码示例使用 Ion 数据类型。

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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一个事务中运行多条语句

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;
    });
};
```

重试逻辑

驱动程序 executeLambda 方法具有内置的重试机制，如果发生可重试的异常(例如超时或 OCC 冲突)，该机制可以重试事务。最大重试次数和退避策略为可配置。

默认重试限值为 4，默认退避策略为 defaultBackoffFunction ，以 10 毫秒为基数。您可以使用 RetryConfig 实例为每个驱动程序实例以及每个事务设置重试配置。

以下代码示例使用自定义重试限制和驱动程序实例的自定义退避策略指定重试逻辑。

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);
```

以下代码示例使用自定义重试限制和自定义回退策略为特定 lambda 执行指定重试逻辑。
此`executeLambda`配置将覆盖为驱动程序实例设置的重试逻辑。

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);
}());
```

实现唯一限制

QLDB 不支持唯一索引，但您可在应用程序中实现此行为。

假设您要对 `Person` 表中的 `GovId` 字段实现唯一性约束。据此，可以编写执行以下操作的事务：

1. 断言该表中没有指定 `GovId` 的现有文档。
2. 如果断言通过，请插入文档。

如果一个竞争事务同时通过断言，则只有一个事务将成功提交。另一笔事务将失败，并显示 OCC 冲突异常。

以下代码示例显示了如何实现此唯一约束。

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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

使用 Amazon Ion

以下各节说明了如何使用 Amazon Ion 模块处理 Ion 数据。

目录

导入 Ion 模块

JavaScript

```
var ionjs = require('ion-js');
```

TypeScript

```
import { dom, dumpBinary, dumpText, load } from "ion-js";
```

创建 Ion 类型

以下代码示例从 Ion 文本创建 Ion 对象。

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']
```

以下代码示例从 Node.js 字典创建 Ion 对象。

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']
```

获取 Ion 二进制转储

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
```

获取 Ion 文本转储

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"}
```

有关 Ion 更多信息，请参阅 GitHub 上的Amazon Ion 文档。有关在 QLDB 中使用 Ion 的更多代码示例，请参阅使用 Amazon QLDB 中的 Amazon Ion 数据类型。

适用于 Python 的 Amazon QLDB 驱动程序

要处理分类账中的数据，您可以使用AWS提供的驱动程序从 Python 应用程序连接到 Amazon QLDB。以下主题介绍了如何开始使用适用于 Python 的 QLDB 驱动程序。

主题

驱动程序资源

有关 Python 驱动程序支持功能的更多信息，请参阅以下资源：

- API 参考：3.x, 2.x
- 驱动程序源代码 (GitHub)
- 示例应用程序源代码 (GitHub)
- Amazon Ion 代码示例

先决条件

开始使用适用于 Python 的 QLDB 驱动程序之前，您必须执行以下操作:

1. 按照 访问 Amazon QLDB 中的 AWS 的设置说明进行操作。这包括以下这些：
    1. 注册AWS。
    2. 创建具有适当 QLDB 权限的用户。
    3. 授权以编程方式访问开发。
2. 从 Python 下载网站下载并安装以下 Python 版本之一：
    - 3.6 或更高版本 - 适用于 Python v3 的 QLDB 驱动程序
    - 3.4 或更高版本 - 适用于 Python v2 的 QLDB 驱动程序

3. 设置您的AWS凭据和默认凭据AWS 区域。有关说明，请参阅 AWS SDK for Python (Boto3) 文档中的 快速入门。

   有关可用区域的完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

接下来，您可下载完整的教程示例应用程序，也可以只在 Python 项目中安装驱动程序并运行短代码示例。

- 要在现有项目中安装 QLDB 驱动程序和 AWS SDK for Python (Boto3)，请继续执行安装。
- 要设置项目并运行演示分类账上基本数据事务的简短代码示例，请参阅 快速入门教程。
- 要在完整的教程示例应用程序中运行更深入的数据和管理 API 操作示例，请参阅 Python 教程。

安装

QLDB 支持以下驱动程序版本及其 Python 依赖项。

驱动程序版本	Python 版本	状态	最新发布日期
2.x	3.4 或更高版本	量产版	2020 年 5 月 7 日
3.x	3.6 或更高版本	量产版	2021 年 10 月 28 日

要使用 `pip`（适用于 Python 的软件包管理器）从 PyPI 安装 QLDB 驱动程序，请在命令行输入以下命令。

3.x

```
pip install pyqldb
```

2.x

```
pip install pyqldb==2.0.2
```

安装驱动程序还会安装其依赖项，包括 AWS SDK for Python (Boto3) 和 Amazon Ion 软件包。

使用驱动程序连接至分类账

然后，您可以导入驱动程序，并使用它来连接到分类账。以下 Python 代码示例说明如何为指定的分类账名称创建会话。

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)
```

有关如何在分类账上运行基本数据事务的简短代码示例，请参阅 说明书参考。

适用于 Python 的 Amazon QLDB 驱动程序 — 快速入门教程

在本教程中，您将学习如何使用适用于 Python 的最新版 Amazon QLDB 驱动程序来设置简单应用程序。本指南包括安装驱动程序的步骤以及创建、读取、更新和删除 (CRUD) 的基本操作的简短代码示例。有关在完整示例应用程序中演示这些操作的更深入的示例，请参阅 Python 教程。

主题

先决条件

在开始之前，请务必执行以下操作：

1. 请为 Python 驱动程序完成 先决条件（如果尚未执行此操作）。这包括注册 AWS、授予开发所需的编程访问权限以及安装 Python 3.6 及以上版本。
2. 创建一个名为 `quick-start` 分类账。

   要了解如何创建分类账，请参阅控制台入门中的 Amazon QLDB 分类账的基本操作 或 第 1 步：创建新分类账。

步骤 1：设置您的 项目

首先，请设置您的 Python 项目。

> **Note**
>
> 如果您使用的 IDE 具有自动执行这些设置步骤的功能，则可以直接跳到 第 2 步：初始化驱动程序。

1. 为应用程序创建一个文件夹。

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

2. 要从 PyPI 安装适用于 Python 的 QLDB 驱动程序，请输入以下 `pip` 命令。

   ```
   $ pip install pyqldb
   ```

   安装驱动程序还会安装其依赖项，包括 AWS SDK for Python (Boto3) 和 Amazon Ion 软件包。

3. 创建名为 `app.py` 的新文件。

   然后，按以下步骤中逐步添加代码示例，尝试一些基本的 CRUD 操作。或者，您可以跳过分步教程，改为运行完整的应用程序。

第 2 步：初始化驱动程序

初始化连接到名为 `quick-start` 的分类账的驱动程序实例。将以下代码添加到您的 `app.py` 文件。

```
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)
```

第 3 步：创建表和索引

以下代码示例显示如何运行 `CREATE TABLE` 和 `CREATE INDEX`。

添加以下代码，创建名为 `People` 的表，并为该表上的 `lastName` 字段创建索引。索引是优化查询性能和帮助限制乐观并发控制 (OCC) 冲突异常所必需的。

```
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))
```

第 4 步：插入文档

以下代码示例显示如何运行 `INSERT` 语句。QLDB 支持 PartiQL 查询语言（兼容 SQL）和 Amazon Ion 数据格式（JSON 的超集）。

添加以下代码，在 `People` 表格中插入文档。

```
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))
```

此示例使用问号 (?) 作为变量占位符，将文档信息传递给语句。该 `execute_statement` 方法同时支持 Amazon Ion 类型和 Python 本机类型。

> **Tip**
>
> 要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个列表类型的参数，如下所示。
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> 传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

第 5 步：查询文档

以下代码示例显示如何运行 SELECT 语句。

添加以下代码，用于从 People 表格中查询文档。

```
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))
```

第 6 步：更新文档

以下代码示例显示如何运行 UPDATE 语句。

1. 添加以下代码，通过更新 age 到 42 来更新 People 表中的文档。

```
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. 再次查询表以查看更新的值。

```
# Query the updated document
qldb_driver.execute_lambda(lambda executor: read_documents(executor))
```

3. 要运行该应用程序，请在项目目录中输入以下命令。

```
$ python app.py
```

运行完整的应用程序

以下代码示例是 app.py 应用程序的完整版本。您还可以从头到尾复制并运行此代码示例，而不必单独执行前面的步骤。此应用程序演示了对名为 quick-start 的分类账的一些基本的 CRUD 操作。

> **Note**
>
> 在运行此代码之前，请确保 quick-start 分类账中还没有名为 People 的活动表。

```
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))
```

要运行完整的应用程序，请在项目目录中输入以下命令。

```
$ python app.py
```

适用于 Python 的Amazon QLDB 驱动程序 — 说明书参考

本参考指南显示了 Python 的 Amazon QLDB 驱动程序的常见用例。它提供了 Python 代码示例，演示了如何使用该驱动程序运行基本的创建、读取、更新和删除 (CRUD) 操作。它还包括用于处理 Amazon Ion 数据的代码示例。此外，本指南还重点介绍了使事务具有幂等性和实现唯一性约束的最佳实践。

> **Note**
>
> 在适用的情况下，某些用例对于 Python 的 QLDB 驱动程序的每个支持主要版本都配备不同代码示例。

目录

导入驱动程序

下面的代码示例导入驱动程序。

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**
>
> 此示例还导入了 Amazon Ion 软件包 (`amazon.ion.simpleion`)。在本参考中运行某些数据操作时，您需要此软件包来处理 Ion 数据。要了解更多信息，请参阅 使用 Amazon Ion。

实例化驱动程序

以下代码示例使用默认设置创建连接到指定分类账名称的驱动程序实例。

3.x

```
qldb_driver = QldbDriver(ledger_name='vehicle-registration')
```

2.x

```
qldb_driver = PooledQldbDriver(ledger_name='vehicle-registration')
```

CRUD 操作

QLDB 作为事务的一部分运行创建、读取、更新和删除 (CRUD) 操作。

> **Warning**
>
> 作为最佳实践，使写事务严格地幂等。

使事务幂等

我们建议将写事务设置为幂等，以避免重试时出现任何意想不到的副作用。如果事务可以运行多次并每次都产生相同的结果，则事务是幂等的。

例如，假设有一个事务，要将文档插入名为 `Person` 的表中。事务应该首先检查文档是否已经存在于表中。如果没有这种检查，表最终可能会有重复的文档。

假设 QLDB 在服务器端成功提交了事务，但客户端在等待响应时超时。如果事务不是幂等的，则在重试的情况下可以多次插入相同的文档。

使用索引避免全表扫描

我们还建议您在索引字段或文档 ID 上使用相等运算符来运行带有`WHERE`谓词子句的语句；例如，`WHERE indexedField = 123`或 `WHERE indexedField IN (456, 789)`。如果没有这种索引查找，QLDB 需要进行表扫描，这可能会导致事务超时或乐观并发控制 (OCC) 冲突。

有关 OCC 的更多信息，请参阅Amazon QLDB 并发模型。

隐式创建的事务

pyqldb.driver.qldb_driver.execute_lambda 方法接受一个 lambda 函数，该函数接收一个 TransactionExecutor 的实例，您可以用它来运行语句。Executor 的实例封装了隐式创建的事务。

您可以使用事务执行器的 execute_statement 法在 lambda 函数中运行语句。当 lambda 函数返回时，驱动程序会隐式提交事务。

> **Note**
>
> 该 execute_statement 方法同时支持 Amazon Ion 类型和 Python 本机类型。如果您将 Python 本地类型作为参数传递给 execute_statement，则驱动程序会使用 amazon.ion.simpleion 模块将其转换为 Ion 类型（前提是支持对给定 Python 数据类型的转换）。有关支持的数据类型和转换规则，请参阅 simpleion 源代码。

以下各节介绍如何运行基本的 CRUD 操作、指定自定义重试逻辑以及如何实现唯一性约束。

目录

创建表

```
def create_table(transaction_executor):
    transaction_executor.execute_statement("CREATE TABLE Person")
```

```
qldb_driver.execute_lambda(lambda executor: create_table(executor))
```

创建索引

```
def create_index(transaction_executor):
    transaction_executor.execute_statement("CREATE INDEX ON Person(GovId)")

qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

阅读文档

```
# 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))
```

使用查询参数

以下代码示例使用原生类型查询参数。

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?",
 'TOYENC486FH')
```

以下代码示例使用 Ion 类型查询参数。

```
name = ion.loads('Brent')
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE FirstName
 = ?", name)
```

以下代码示例使用了多个查询参数。

```
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", 'TOYENC486FH', "Brent")
```

以下代码示例使用查询参数列表。

```
gov_ids = ['TOYENC486FH','ROEE1','YH844']
cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", *gov_ids)
```

> **Note**
>
> 当您在没有索引查找的情况下运行查询时，它会调用全表扫描。在此示例中，我们建议在GovId字段上设置 索引以优化性能。如果不开启GovId索引，查询可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

插入文档

以下代码示例插入本地数据类型。

```
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))
```

以下代码示例插入 Ion 数据类型。

```
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))
```

此事务将文档插入 Person 表中。在插入之前，它首先检查文档是否已存在于表格内。此检查使事务本质上是幂等。即使您多次运行此事务，也不会造成任何异常副作用。

> **Note**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一条语句内插入多个文档

要使用单个 INSERT 语句插入多个文档，可以向该语句传递一个列表类型的参数，如下所示。

```
# people is a list
transaction_executor.execute_statement("INSERT INTO Person ?", people)
```

传递 Ion 列表时，不要将变量占位符 (?) 括在双尖括号 (<<...>>) 内。在手动 PartiQL 语句中，双尖括号表示名为bag的无序集合。

更新文档

以下代码示例使用原生数据类型。

```
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))
```

以下代码示例使用 Ion 数据类型。

```
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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

删除文档

以下代码示例使用原生数据类型。

```
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))
```

以下代码示例使用 Ion 数据类型。

```
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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

在一个事务中运行多条语句

```
# 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))
```

重试逻辑

驱动程序 `execute_lambda` 方法具有内置的重试机制，如果发生可重试的异常(例如超时或 OCC 冲突)，该机制可以重试事务。

3.x

最大重试次数和退避策略为可配置。

默认的重试限制为 4，默认的退避策略是以10毫秒为基数的 Exponential Backoff and Jitter。您可以使用RetryPolicy实例为每个驱动程序实例以及每个事务设置重试策略。

以下代码示例使用自定义重试限制和驱动程序实例的自定义退避策略指定重试逻辑。

```
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)
```

以下代码示例使用自定义重试限制和自定义回退策略为特定 lambda 执行指定重试逻辑。此execute_lambda配置将覆盖为驱动程序实例设置的重试逻辑。

```
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

最大重试次数。您可以通过在初始化PooledQldbDriver 时设置 retry_limit 属性来配置重试限制。

默认重试限制为 4。

实现唯一限制

QLDB 不支持唯一索引，但您可在应用程序中实现此行为。

假设您要对 Person 表中的 GovId 字段实现唯一性约束。据此，可以编写执行以下操作的事务：

1. 断言该表中没有指定 GovId 的现有文档。
2. 如果断言通过，请插入文档。

如果一个竞争事务同时通过断言，则只有一个事务将成功提交。另一笔事务将失败，并显示 OCC 冲突异常。

以下代码示例显示了如何实现此唯一约束。

```
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**
>
> 在此示例中，我们建议在 GovId 字段上设置索引以优化性能。如果不开启GovId索引，语句可能会有更长的延迟，还可能导致 OCC 冲突异常或者事务超时。

使用 Amazon Ion

以下各节说明了如何使用 Amazon Ion 模块处理 Ion 数据。

目录

导入 Ion 模块

```
import amazon.ion.simpleion as simpleion
```

创建 Ion 类型

以下代码示例从 Ion 文本创建 Ion 对象。

```
ion_text = '{GovId: "TOYENC486FH", FirstName: "Brent"}'
ion_obj = simpleion.loads(ion_text)

print(ion_obj['GovId']) # prints TOYENC486FH
print(ion_obj['Name']) # prints Brent
```

以下代码示例从 Python `dict`创建 Ion 对象。

```
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
```

获取 Ion 二进制转储

```
# 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'
```

获取 Ion 文本转储

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj, binary=False)) # prints $ion_1_0
 {GovId:'TOYENC486FH',FirstName:"Brent"}
```

有关使用 Ion 的更多信息，请参阅 GitHub 上的 Amazon Ion 文档。有关在 QLDB 中使用 Ion 的更多代码示例，请参阅使用 Amazon QLDB 中的 Amazon Ion 数据类型。

在 Amazon QLDB 中通过驱动程序了解会话管理

如您有使用关系数据库管理系统 (RDBMS) 的经验，可能已熟悉并发连接。QLDB 与传统 RDBMS 连接的概念不同，因为事务通过 HTTP 请求和响应消息运行。

在 QLDB 中，类比概念是活跃会话。从概念上讲，会话类似于用户登录，它管理有关您对分类账的数据事务请求的信息。活动会话是指正在积极运行事务会话。它可以是最近完成事务的会话，服务预计它将立即启动另一笔事务。QLDB 支持每个会话正在运行的事务。

每个分类账的并发活动会话限制在Amazon QLDB 资源中的限额和限制中定义。达到此限制后，任何尝试启动事务的会话都会导致错误 (`LimitExceededException`)。

有关使用 QLDB 驱动程序在应用程序中配置会话池的最佳实践，请参阅Amazon QLDB 驱动程序建议中的配置 QldbDriver 对象。

目录

- 会话池概述
- 会话池与事务逻辑
- 将会话重返数据池

会话生命周期

以下 QLDB 会话 API 操作序列代表了 QLDB 会话的典型生命周期：

1. `StartSession`
2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. 重复步骤 2—4 以启动更多事务（一次一个事务）。
6. `EndSession`

会话过期

无论会话是否在积极运行事务，QLDB 都会在总生命周期为13–17 分钟后过期并丢弃会话。会话丢失或受损的原因有很多，如硬件故障、网络故障或应用程序重启。QLDB 对会话强制使用最长生命周期，以确保客户端应用程序能应对会话故障。

QLDB 驱动程序中的会话处理

尽管您可以使用 AWS SDK 直接与 QLDB 会话 API 进行交互，但这会增加复杂性，需要您计算提交摘要。QLDB 使用此提交摘要确保事务完整性。我们建议不要直接与 API 交互，而是使用 QLDB 驱动程序。

该驱动程序在事务数据 API 中提供了高级抽象层。它通过管理 SendCommand API 调用，简化了对分类账数据运行 PartiQL 语句的过程。这些 API 调用需要驱动程序为您处理的多项参数，包括会话管理、事务管理以及出现错误时的重试策略。

主题

- 会话池概述
- 会话池与事务逻辑

- 将会话重返数据池

会话池概述

在旧版本的 QLDB 驱动程序 (例如 Java v1.1.0) 中，我们提供了驱动对象的两种实现：标准非池化QldbDriver和 PooledQldbDriver。顾名思义，PooledQldbDriver 维护着一个可在事务中重复使用的会话池。

根据用户反馈，开发人员更喜欢默认使用池化功能及其优势，而非使用标准驱动程序。因此，我们删除了PooledQldbDriver并将会话池功能移至QldbDriver。此更改包含在每个驱动程序的最新版本中 (例如 Java v2.0.0)。

该驱动程序提供了三种抽象级别：

- 驱动程序(池化驱动程序实现) - 顶级抽象。驱动程序维护和管理会话池。当您要求驱动程序运行事务时，驱动程序会从池中选择会话并使用它来运行事务。如果事务由于会话错误 (InvalidSessionException) 而失败，则驱动程序将使用另一会话重试该事务。从本质上讲，该驱动程序提供了完全托管会话体验。
- 会话 - 比驱动程序抽象低一个级别。会话包含在池内，驱动程序管理会话的生命周期。如果事务失败，驱动程序会使用同一个会话、以最多指定次数重试该事务。但是，如果会话遇到错误 (InvalidSessionException)，QLDB 将在内部将其丢弃。然后，驱动程序负责从池中获取另一会话以重试该事务。
- 事务 - 最低的抽象级别。事务包含在会话中，会话则管理事务的生命周期。如果出现错误，会话负责重试事务。该会话还确保它不泄露尚未提交或取消的未完成事务。

在每个驱动程序的最新版本，您只能在驱动程序抽象级别执行操作。您无法直接控制单个会话和事务 (也就是说，没有用于手动启动新会话或事务的 API 操作)。

会话池与事务逻辑

驱动程序的最新版本不再提供驱动程序对象的非池化实现。默认情况下，该 QldbDriver 对象管理会话池。当您调用运行事务时，驱动程序将执行以下步骤：

1. 驱动程序检查其是否达到会话池限制。如果是，驱动程序会立即抛出 NoSessionAvailable 异常。否则，转到下一步。
2. 驱动程序检查池中是否有可用会话。
    - 如果池中存在可用会话，则驱动程序将使用该会话来运行事务。

- 如果池中没有可用会话，则驱动程序将创建新会话并使用该会话来运行事务。

3. 在第 2 步中驱动程序获得会话后，驱动程序将在会话实例上调用该 `execute` 操作。
4. 在会话的 `execute` 操作中，驱动程序尝试通过调用`startTransaction`来启动事务。
   - 如果会话无效，则`startTransaction`调用失败，驱动程序将返回至步骤 1。
   - 如果 `startTransaction` 调用成功，驱动程序将继续下一步。
5. 驱动程序运行您的 lambda 表达式。此 lambda 表达式可包含一个或多个用于运行 PartiQL 语句的调用。在 lambda 表达式完成运行且没有任何错误后，驱动程序将继续提交事务。
6. 事务的提交结果如下：
   - 提交成功，驱动程序将控制权返回至您的应用程序代码。
   - 因乐观并发控制 (OCC) 冲突，提交失败。在这种情况下，驱动程序使用相同会话重试步骤 4—6。最大重试次数可以在应用程序代码中配置。默认限制为 4。

> **Note**
>
> 如果`InvalidSessionException`在步骤 4—6 期间抛出，则驱动程序会将会话标记为已关闭，然后返回步骤 1 重试事务。
> 如果在步骤 4—6 中引发任何其他异常，则驱动程序会检查是否可以重试该异常。如果是这样，它会重试此事务，直到达到指定的重试次数。否则，它会将异常传播（冒泡并抛出）至您的应用程序代码中。

将会话重返数据池

如果在活动事务过程中的任何时候发生`InvalidSessionException`，则驱动程序不会将会话返回到池中。QLDB 改为丢弃会话，驱动程序会从池内获得另一个会话。在所有其它情况下，驱动程序会将会话返回池中。

Amazon QLDB 驱动程序建议

本部分介绍为任何支持的语言配置和使用 Amazon QLDB 驱动程序的最佳实践标准。所提供的代码示例专门针对 Java。

这些建议适用于大多数典型用例，但一种方法并不适合所有情况。使用您认为适合您的应用程序的以下建议。

主题

配置 QldbDriver 对象

该QldbDriver对象通过维护可跨事务重复使用的会话池来管理与分类账的连接。会话代表与分类账的单个连接。QLDB 支持每个会话正在运行的事务。

Important

对于较旧的驱动程序版本，会话池功能仍位于PooledQldbDriver对象中，而非QldbDriver。如果您使用的是以下版本之一，请在本主题的其余部分中将任何提及的 QldbDriver 内容替换为PooledQldbDriver。

驱动程序	版本
Java	1.1.0 或更早
.NET	0.1.0-beta
Node.js	1.0.0-rc.1 或更早
Python	2.0.2 或更早

最新版本的驱动中已弃用该 PooledQldbDriver对象。建议升级至最新版本，并将的所有实例转换 PooledQldbDriver 为 QldbDriver。

将 QldbDriver 配置为全局对象

若优化驱动程序和会话的使用，请确保您的应用程序实例中仅存在一个驱动程序的全局实例。例如，在 Java 中，你可以使用依赖注入框架，例如 Spring、Google Guice 或Dagger。以下代码示例显示如何配置 QldbDriver 单例。

```
@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();
}
```

配置重试次数

当出现常见的瞬态异常 (例如SocketTimeoutException或NoHttpResponseException) 时，驱动程序会自动重试事务。要设置最大重试次数，可以在创建的实例时使用 transactionRetryPolicy 配置对象的 maxRetriesQldbDriver 参数。(对于上一节中列出的较旧驱动程序版本，请使用PooledQldbDriver的参数retryLimit。)

maxRetries 的默认值为 4。

客户端错误，例如 “InvalidParameterException 无法重试”。当它们发生时，事务将中止，会话返回至池中，并将异常抛给驱动程序的客户端。

配置最大并行会话与事务数

QldbDriver的实例用于运行事务的最大分类账会话数由其maxConcurrentTransactions参数定义。(对于上一节中列出的较旧的驱动程序版本，这是由PooledQldbDriver的poolLimit参数定义的。)

此限制必须大于零且小于或等于会话客户端允许的最大打开 HTTP 连接数 由特定 AWS SDK 定义)。例如，在 Java 中，最大连接数是在ClientConfiguration 对象中设置的。

maxConcurrentTransactions的默认值是您的 AWS SDK 的最大连接设置。

在应用程序 QldbDriver 中配置时，请考虑以下扩展注意事项：

- 您的池中的会话数应始终至少与您计划同时运行的事务数量一样多。
- 在监督线程委托给工作线程的多线程模型中，驱动程序应至少拥有与工作线程数量一样多的会话。否则，在峰值负载时，线程将排队等待可用会话。
- 每个分类账的并发活动会话的服务限制在Amazon QLDB 资源中的限额和限制中定义。确保您配置的并发会话数不超过此限制，以便用于所有客户端单个分类账。

重试异常

重试 QLDB 事出现的异常时，请考虑以下建议。

重试 OccConflictExcept

当事务访问的数据自事务开始以来发生更改时，就会发生乐观并发控制 (OCC) 冲突异常。QLDB 在尝试提交事务时抛出此异常结果。驱动程序最多会根据配置重试事务`maxRetries`次数。

有关 OCC 的更多信息以及使用索引限制 OCC 冲突的最佳实践，请参阅Amazon QLDB 并发模型。

在 QldbDriver 之外重试其他异常

要在运行时抛出自定义的应用程序定义的异常时在驱动程序外部重试事务，您必须包装该事务。例如，在 Java 中，以下代码显示了如何使用 Reslience4J 库在 QLDB 内重试事务。

```
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**
>
> 在 QLDB 驱动程序之外重试事务会使其产生乘数效应。例如，如果配置`QldbDriver`为重试三次，并且自定义重试逻辑也重试三次，则同一事务最多可以重试九次。

使事务幂等

我们建议将写事务设置为幂等，以避免重试时出现任何意想不到的副作用。如果事务可以运行多次并每次都产生相同的结果，则事务是幂等的。

要了解更多信息，请参阅 Amazon QLDB 并发模型。

优化性能

要在使用驱动程序运行事务时优化此性能，请考虑以下注意事项：

- 该`execute`操作始终对 QLDB 进行至少三次 SendCommand API 调用，包含以下命令：
  1. `StartTransaction`
  2. `ExecuteStatement`

     对于您在`execute`数据块中运行的每条 PartiQL 语句，都会调用此命令。
  3. `CommitTransaction`

  在计算应用程序的总体工作负载时，请考虑发出 API 调用的总数。
- 一般来说，我们建议从单线程编写器开始，并通过在单个事务中批处理多个语句来优化事务。最大限度地提高事务大小、文档大小和每笔事务的文档数量的限额，如Amazon QLDB 资源中的限额和限制中所定义。
- 如果批处理不足以满足大型事务负载，您可以通过添加其他编写器来尝试多线程。但是，您应该仔细考虑您的应用程序对文档和事务排序的要求，以及由此带来的额外复杂性。

每笔事务运行多个语句

如上一节所述，您可为每个事务运行多个语句以优化应用程序的性能。在以下代码示例中，您可查询到一个表，然后在事务中更新该表中的文档。您可以通过向execute操作传递 lambda 表达式来实现此目的。

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))
```

驱动程序的 `execute` 操作会隐式启动会话和该会话中的事务。您在 lambda 表达式中运行的每条语句都包含在此事务中。所有语句运行后，驱动程序将自动提交事务。如果在自动重试限制用尽后任何语句失败，则事务将会中止。

在事务中传播异常

当每个事务运行多个语句时，我们通常不建议您捕获并吞掉事务内的异常。

例如，在 Java 中，以下程序会捕获的任何`RuntimeException`实例，记录错误并继续。此代码示例被认为是错误做法，因为即使`UPDATE`语句失败，事务也会成功。因此，客户端可能会假定更新成功，而更新却没有成功。

> **Warning**
>
> 切勿使用此代码示例。提供它是为了展示被认为是不良做法的反模式示例。

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

改为传播（冒泡）异常。如果事务的任何部分失败，则让`execute`操作中止事务，以便客户端可以相应地处理异常。

使用 Amazon QLDB 中的驱动程序了解重试策略

Amazon QLDB 驱动程序使用重试策略，通过透明重试失败的事务来处理暂时异常。这些异常 (例如`CapacityExceededException`和`RateExceededException`) 通常会在一段时间后自行纠正。如果在适当延迟之后重试因异常而失败的事务，则很可能会成功。这有助于提高使用 QLDB 应用程序的稳定性。

主题

- 可重试错误类型
- 默认重试策略

可重试错误类型

当且仅当该事务的操作期间出现以下任何异常时，驱动程序才会自动重试此事务：

- CapacityExceededException – 当请求超过分类账的处理能力时返回。
- InvalidSessionException – 当会话不再有效或会话不存在时返回。
- LimitExceededException – 如果超过资源限制(例如活动会话数)，则返回。
- OccConflictException – 当由于乐观并发控制(OCC) 的验证阶段失败而导致事务无法写入日志时返回。
- RateExceededException – 请求速率超出允许吞吐量允许吞吐量时返回。

默认重试策略

重试策略由重试条件与退避策略组成。重试条件定义何时应重试事务，而退避策略定义在重试事务之前的等待时间。

创建驱动程序实例时，默认重试策略指定最多重试四次，并使用指数退避策略。指数退避策略使用最小延迟 10 毫秒，最大延迟 5000 毫秒，抖动相等。如果无法在重试策略内成功提交此事务，我们建议您改时尝试该事务。

指数回退的原理是对于连续错误响应，重试等待间隔越来越长。有关更多信息，请参阅 AWS 博文 指数回退和抖动。

Amazon QLDB 驱动程序中的常见错误

本节描述了 Amazon QLDB 驱动程序在与 QLDB 会话 API 交互时可能引发的运行时错误。

下面是驱动程序返回的常见异常列表。每个异常都包括特定的错误消息，然后是简短的描述和可能的解决方案建议。

CapacityExceededException

消息：容量已超出

Amazon QLDB 拒绝了该请求，因为它超出了分类账的处理能力。QLDB 对每个分类账强制执行内部扩展限制，以维护服务的运行状况和性能。此限制因每个请求的工作负载大小而异。例如，如果请求执行效率低下的数据事务，例如由非索引限定查询产生的表扫描，则该请求的工作负载可能会增加。

我们建议您在重试请求之前等待。如果您的应用程序一直遇到此异常，请优化您的语句并降低发送到分类账的请求的速率和数量。语句优化的示例包括每个事务运行更少的语句和调优表索引。要了解如何优化语句和避免表扫描，请参阅 优化查询性能。

建议使用最新版本的 QLDB 驱动程序。驱动程序具有默认的重试策略，该策略使用指数回退和抖动来自动重试此类异常。指数回退的原理是对于连续错误响应，重试等待间隔越来越长。

InvalidSessionException

消息：事务 *transactionId* 已过期

事务已超过其最长生命周期。在提交之前，事务最多可运行 30 秒。超过此时间限制后，QLDB 会拒绝在事务完成的任何工作，并丢弃运行该事务会话。此限制通过启动事务（而不提交或取消事务）以保护客户端免遭泄漏会话。

如果这是应用程序中常见的异常，那么很可能是事务运行时间过长。如果事务运行时间超过 30 秒，请优化您的语句以加快事务速度。语句优化的示例包括每个事务运行更少的语句和调优表索引。有关更多信息，请参阅优化查询性能。

InvalidSessionException

消息：会话 *sessionId* 已过期

QLDB 放弃了会话，因为它已超过其最大总生命周期。无论事务是否处于活动状态，QLDB 都会在 13-17 分钟后丢弃会话。会话丢失或受损的原因有很多，如硬件故障、网络故障或应用程序重启。因此，QLDB 对会话强制使用最长生命周期，以确保客户端软件能应对会话故障。

如果您遇到此异常，我们建议您获取一个新的会话并重试该事务。我们还建议使用最新版本的 QLDB 驱动程序，该驱动程序代表应用程序管理会话池及其运行状况。

InvalidSessionException

消息：没有这样的会话

客户试图使用不存在的会话与 QLDB 进行事务处理。假设客户端正在使用先前存在的会话，则由于以下原因之一，该会话可能不再存在：

- 如果会话涉及内部服务器故障（即 HTTP 响应代码 500 错误），那么 QLDB 可能会选择完全放弃该会话，而不是允许客户使用状态不确定的会话进行事务。然后，对该会话的任何重试尝试都会失败，并出现此错误。
- 过期的会话最终会被 QLDB 遗忘。然后，任何继续使用会话的尝试都会导致此错误，而不是初始 `InvalidSessionException`。

如果您遇到此异常，我们建议您获取一个新的会话并重试该事务。我们还建议使用最新版本的 QLDB 驱动程序，该驱动程序代表应用程序管理会话池及其运行状况。

RateExceededException

消息：已超出速率

QLDB 根据调用者的标识对客户端进行节流。QLDB 使用令牌桶节流算法在每个区域、每个账户的基础上强制执行节流。QLDB 这样做是为了帮助提高服务的性能，并确保所有 QLDB 客户的公平使用。例如，尝试使用 `StartSessionRequest` 操作获取大量并发会话可能会导致节流。

为了保持应用程序的运行状况并缓解进一步的节流，您可以使用指数回退和抖动来重试此异常。指数回退的原理是对于连续错误响应，重试等待间隔越来越长。建议使用最新版本的 QLDB 驱动程序。驱动程序具有默认的重试策略，该策略使用指数回退和抖动来自动重试此类异常。

如果您的应用程序持续受到 QLDB 的 `StartSessionRequest` 调用节流，那么最新版本的 QLDB 驱动程序也可以提供帮助。驱动程序维护着一个跨事务重复使用的会话池，这有助于减少应用程序发出的 `StartSessionRequest` 调用次数。要请求提高 API 节流限额，请联系 AWS Support 中心。

LimitExceededException

消息：已超出会话限制

分类账的活跃会话数量超过了其配额（也称为限制）。此配额在 Amazon QLDB 资源中的限额和限制 中定义。分类账的活动会话计数最终是一致的，并且始终在配额附近运行的分类账可能会定期看到此异常。

为了保持应用程序的运行状况，我们建议重试此异常。为避免出现这种异常，请确保在所有客户端上为单个分类账配置的并发会话不超过 1,500 个。例如，您可以使用适用于 Java 的 Amazon QLDB 驱动程序的 maxConcurrentTransactions 方法来配置驱动程序实例中的最大可用会话数。

QldbClientException

消息：流结果仅在父事务打开时有效

事务已关闭，无法用于从 QLDB 检索结果。事务在提交或取消时关闭。

当客户端直接处理 `Transaction` 对象，并且在提交或取消事务后尝试从 QLDB 检索结果时，就会发生此异常。为了缓解此问题，客户端必须在关闭事务之前读取数据。

使用示例应用程序教程，开始使用 Amazon QLDB

在本教程中，您将使用 Amazon QLDB 驱动程序和软件开发工具包创建 QLDB 分类账，并在AWS中填充示例数据。该驱动程序允许您的应用程序通过事务数据 API 与 QLDB 交互。该 AWS 软件开发工具包支持与 QLDB 资源管理 API 交互。

您在此场景中创建的示例分类账是一个机动车辆部门 (DMV) 数据库，用于追踪有关车辆登记的完整历史信息。以下主题说明了如何添加车辆登记、修改车辆登记，以及如何查看这些登记的更改历史记录。本指南还向您展示了如何以加密方式验证注册文档，最后还会清理资源并删除示例分类账。

示例应用程序可提供以下编程语言版本。

主题

- Amazon QLDB Java 教程

- Amazon QLDB Node.js 教程
- Amazon QLDB Python 教程

Amazon QLDB Java 教程

在本教程样例应用程序的实现中，您将使用 Amazon QLDB 驱动程序通过 AWS SDK for Java 来创建 QLDB 分类账，并用样例数据填充它。

在学习本教程时，您可以参考 AWS SDK for Java API 参考 以了解管理 API 操作。有关事务数据操作，您可参见适用于 Java 的 QLDB 驱动程序 API 参考。

> **Note**
>
> 在适用的情况下，某些用例对于 Java 的 QLDB 驱动程序的每个支持主要版本都配备不同的命令或代码示例。

主题

安装 Amazon QLDB Java 示例应用程序

本节介绍如何按分步 Java 教程，安装和运行 Amazon QLDB 示例应用程序。此示例应用程序的用例是机动车辆部门 (DMV) 数据库，用于追踪有关车辆登记的完整历史信息。

适用于 Java 的 DMV 示例应用程序参见 GitHub 存储库aws-samples/amazon-qldb-dmv-sample-java。

先决条件

在开始之前，请确保您已完成适用于 Java先决条件 的 QLDB 驱动程序。这包括以下这些：

1. 注册AWS。
2. 创建具有适当 QLDB 权限的用户。要完成本教程中的所有步骤，您需要通过 QLDB API 对分类账资源拥有完全管理权限。
3. 如果您使用的是AWS Cloud9以外的 IDE，请安装 Java 并授予开发所需的编程访问权限。

安装

以下步骤介绍如何在本地开发环境中下载与设置示例应用程序。或者，您可以使用AWS Cloud9作为 IDE 自动设置示例应用程序，并使用AWS CloudFormation模板配置开发资源。

本地开发环境

这些说明描述了如何利用自有资源和开发环境下载和安装 QLDB Java 示例应用程序。

下载并运行示例应用程序

1. 输入以下命令，以从 GitHub 克隆示例应用程序。

   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
   ```

   此软件包包含来自Java 教程的 Gradle 配置和完整代码。
2. 下载并运行提供的应用程序。
   - 如果您使用的是 Eclipse：
     a. 启动 Eclipse，然后在Eclipse菜单选择文件、导入，然后选择 现有 Gradle 项目。
     b. 在项目根目录中，浏览并选择包含 `build.gradle` 文件的应用程序目录。然后，选择完成 以使用默认 Gradle 设置进行导入。

c. 你可以尝试运行 `ListLedgers` 程序作为示例。打开 `ListLedgers.java` 文件的上下文菜单 (右键单击)，选择 作为 Java 应用程序运行。

- 如果您使用的是 IntelliJ：

a. 启动 IntelliJ，在 IntelliJ 菜单选择 文件，然后选择 打开。

b. 在项目根目录中，浏览并选择包含 `build.gradle`文件的应用程序目录。然后选择 OK（确定）。保留默认设置，然后再次选择 确定。

c. 你可以尝试运行 `ListLedgers` 程序作为示例。打开 `ListLedgers.java` 文件的上下文菜单 (右键单击)，选择 运行 'ListLedgers'。

3. 继续 步骤 1：创建新分类账 开始教程并创建分类账。

AWS Cloud9

这些说明描述了如何使用AWS Cloud9作为 IDE 自动设置适用于 Java 的 Amazon QLDB 车辆登记示例应用程序。在本指南中，您将使 用AWS CloudFormation 模板配置您的开发资源。

有关 AWS Cloud9 的更多信息，请参阅《AWS Cloud9 用户指南》。要了解有关 AWS CloudFormation 的更多信息，请参阅 AWS CloudFormation 用户指南。

主题

- 第 1 部分：配置资源
- 部分 2：设置 IDE
- 第 3 部分：运行 QLDB DMV 示例应用程序

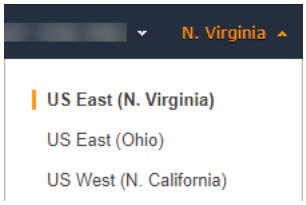
第 1 部分：配置资源

在第一步中，您将使用 AWS CloudFormation Amazon QLDB 示例应用程序预配置设置开发环境所需资源。

打开 AWS CloudFormation 控制台并加载 QLDB 示例应用程序模板

1. 登录到 AWS Management Console 并打开 AWS CloudFormation 控制台 https://console.aws.amazon.com/cloudformation。

   切换至支持 QLDB 的区域。有关完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。以下AWS Management Console屏幕截图按选定AWS 区域显示美国东部（弗吉尼亚州北部）。



2. 在AWS CloudFormation控制台，选择创建堆栈，然后选择使用新资源(标准)。
3. 在 Create stack (创建堆栈) 页面上，选择 指定模板，选择 Amazon S3 URL。
4. 输入以下 URL，然后选择 下一步。

```
https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
```

5. 输入堆栈名称 (例如**qldb-sample-app**)，然后选择下一步。
6. 您可以根据需要添加任何标签，并保留默认选项。然后选择下一步。
7. 检查您的堆栈设置，然后选择创建堆栈。此AWS CloudFormation 脚本可能需要几分钟才能完成。

   此脚为您的AWS Cloud9环境配备关联的 Amazon Elastic Compute Cloud (Amazon EC2) 实例置备，您可使用该实例运行本教程中的 QLDB 示例应用程序。它还会将aws-samples/amazon-qldb-dmv-sample-java存储库从 GitHub 克隆至您的AWS Cloud9开发环境。

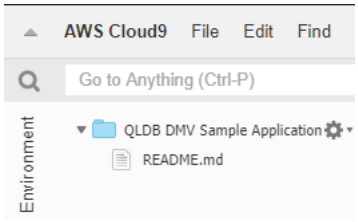
部分 2：设置 IDE

在此步骤中，您已完成云开发环境设置。您可以下载并运行所提供的 Shell 脚本，使用示例应用程序的依赖项设置 AWS Cloud9 IDE。

设置 AWS Cloud9 环境

1. 打开 AWS Cloud9 控制台，地址：https://console.aws.amazon.com/cloud9/。
2. 在 您的环境，定位名为 QLDB DMV Sample Application的环境牌，然后选择 打开 IDE。当基础 EC2 实例启动时，您的环境可能需要一分钟才能加载。

   您的 AWS Cloud9 环境已预先配置了运行本教程所需的系统依赖项。在控制台的 环境 导航窗格中，确认您看到一个名为QLDB DMV Sample Application的文件夹。AWS Cloud9 控制台的以下屏幕截图显示了 QLDB DMV 示例应用程序环境文件夹窗格。



如果您没有看到导航窗格，请切换主机左侧的 环境 选项卡。如果您在窗格中看不到任何文件夹，请使用设置图标

()

启用显示环境根目录。

3. 在控制台底部窗格中，您应该会看到一个打开的bash终端窗口。如果您没有看到这个，请从主机顶部的 窗口 菜单中选择新建终端。
4. 接下来，下载并运行安装脚本以安装 OpenJDK 8，如果适用，请从 Git 存储库中查看相应的分支。在上一步创建的AWS Cloud9终端中，依次运行以下两个命令：

   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
   ```

   完成后，您可看到终端中打印了以下消息：

   ```
   ** DMV Sample App setup completed , enjoy!! **
   ```

5. 抽出时间浏览中的示例应用程序代码AWS Cloud9，尤其是在以下目录路径中：`src/main/java/software/amazon/qldb/tutorial`。

第 3 部分：运行 QLDB DMV 示例应用程序

在本步骤中，您将学习如何使用AWS Cloud9运行 Amazon QLDB DMV 示例应用程序任务。要运行示例代码，请返回AWS Cloud9终端或按照第 2 部分：设置 IDE 所述创建一个新终端窗口。

运行示例应用程序

1. 在终端中运行以下命令，以切换至项目根目录：

```
cd ~/environment/amazon-qldb-dmv-sample-java
```

确保您在以下目录路径运行示例。

```
/home/ec2-user/environment/amazon-qldb-dmv-sample-java/
```

2. 以下命令显示了运行每项任务的 Gradle 语法。

```
./gradlew run -Dtutorial=Task
```

例如，运行以下命令，以列出您所在AWS 账户区域的所有分类账。

```
./gradlew run -Dtutorial=ListLedgers
```

3. 继续 步骤 1：创建新分类账 开始教程并创建分类账。
4. （可选）完成教程后，如果您不再需要 AWS CloudFormation 资源，就可以清理它们了。
   a. 打开AWS CloudFormation控制台：https://console.aws.amazon.com/cloudformation，然后删除您在第 1 部分：配置资源中创建的堆栈。
   b. 同时删除 AWS CloudFormation 模板为您创建的 AWS Cloud9 堆栈。

步骤 1：创建新分类账

在此步骤中，您将创建一个名为 `vehicle-registration` 的新 Amazon QLDB 分类账。

创建新的分类账

1. 查看以下文件 (`Constants.java`)，其包含本教程中所有其他程序使用的常量值。

   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**
>
> 对于 Amazon Ion 软件包，您必须在应用程序中使用命名空间`com.amazon.ion`。AWS SDK for Java 依赖于命名空间`software.amazon.ion`下的另一个 Ion 包，但这是一个与 QLDB 驱动程序不兼容的旧包。

```
/*
 * 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**
>
> 该 Constants 类包括开源 Jackson IonValueMapper类实例。在执行读写事务时，您可以使用此映射器处理您的 Amazon Ion 数据。

该 CreateLedger.java 文件还依赖于以下程序 (DescribeLedger.java)，该程序描述了分类账的当前状态。

```
/*
 * 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. 编译并运行 CreateLedger.java 程序以创建名为vehicle-registration的分类账。

   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**
>
> - 在 `createLedger` 调用中，您必须指定分类账名称和权限模式。我们建议使用 STANDARD 权限模式来最大限度地提高分类账数据的安全性。
> - 创建分类账时，将默认启用 删除保护。QLDB 中有一项功能，可防止分类账被任何用户删除。您可以选择使用 QLDB API 或 AWS Command Line Interface (AWS CLI) 在创建分类账时禁用删除保护。

- 您还可选择指定要附加到分类账的标签。

要验证您与新分类账的连接，请继续 步骤 2：测试分类账的连接性。

步骤 2：测试分类账的连接性

在此步骤中，您将验证是否可以使用交易数据 API 端点连接至 Amazon QLDB 中的`vehicle-registration`分类账。

测试分类账的连接性

1. 查看以下程序 (`ConnectToLedger.java`)，该程序创建了与`vehicle-registration` 分类账的数据会话连接。

   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

- 若对分类账运行数据操作，您必须创建`QldbDriver`类实例以连接至特定分类账。这与您在上一步中创建分类账时使用的`AmazonQLDB` 客户端对象不同。此前客户端仅适用于运行Amazon QLDB API 参考中列出的管理 API 操作。
- 首先，创建一个 `QldbDriver` 对象。当创建此驱动程序时，必须指定分类账名称。

  然后，您可以使用此驱动程序的 `execute` 方法运行 PartiQL 语句。
- 或者，您可指定事务异常的最大重试次数。该 `execute` 方法会自动重试乐观并发控制 (OCC) 冲突和其他常见的瞬态异常，但不超过此可配置限制。默认值为 4。

  如果在达到限制后事务仍失败，则驱动程序会抛出异常。要了解更多信息，请参阅使用 Amazon QLDB 中的驱动程序了解重试策略。

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**
>
> - 要对分类账运行数据操作，您必须创建PooledQldbDriver 或 QldbDriver 类的实例以连接到特定分类账。这与您在上一步中创建分类账时使用的 AmazonQLDB 客户端对象不同。之前的客户端仅用于Amazon QLDB API 参考中列出的管理 API 操作。
>
>   除非您需要使用QldbDriver实现自定义会话池，否则我们建议使用PooledQldbDriver。默认PooledQldbDriver池大小是会话客户端允许的最大打开 HTTP 连接数。
> - 首先，创建一个 PooledQldbDriver 对象。创建此驱动程序时，您必须指定分类账名称。

然后，您可以使用此驱动程序的 `execute` 方法运行 PartiQL 语句。或者，您可通过此池驱动程序对象手动创建会话并使用该会话`execute`方法。会话代表与分类账的单个连接。

- 或者，您可指定事务异常的最大重试次数。该 `execute` 方法会自动重试乐观并发控制 (OCC) 冲突和其他常见的瞬态异常，但不超过此可配置限制。默认值为 4。

  如果在达到限制后事务仍失败，则驱动程序会抛出异常。要了解更多信息，请参阅使用 Amazon QLDB 中的驱动程序了解重试策略。

2. 编译并运行该 `ConnectToLedger.java` 程序，以测试您的数据会话与`vehicle-registration` 分类账的连接。

覆盖 AWS 区域

示例应用程序以AWS 区域 默认方式连接至 QLDB，您可以按照先决条件步骤中的说明进行设置设置您的默认 AWS 凭证和区域。可以修改 QLDB 会话客户构建程序的属性来更改区域。

2.x

下面的代码示例实例化一个新的 `QldbSessionClientBuilder` 对象。

```
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);
```

您可以使用 `region` 方法对任何可用区域中的 QLDB 运行您的代码。有关完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

1.x

下面的代码示例实例化一个新的 `AmazonQLDBSessionClientBuilder` 对象。

```
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);
```

您可以使用 withRegion 方法对任何可用区域中的 QLDB 运行您的代码。有关完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

要在 vehicle-registration 分类账中创建表，请继续 步骤 3：创建表、索引与示例数据。

步骤 3：创建表、索引与示例数据

当您的 Amazon QLDB 分类账处于活动状态且接受连接时，您可开始创建有关车辆、车主和注册信息的数据表。创建表和索引后，可以向其中加载数据。

在此步骤中，您将在vehicle-registration分类账中创建四个表格：

- VehicleRegistration
- Vehicle
- Person
- DriversLicense

您还可创建以下索引。

表名称	Field
VehicleRegistration	VIN
VehicleRegistration	LicensePlateNumber
Vehicle	VIN
Person	GovId
DriversLicense	LicenseNumber
DriversLicense	PersonId

插入示例数据时，首先要在 Person 表格中插入文档。然后使用系统分配的、来自每个Person文档的id填充适当VehicleRegistration 和 DriversLicense文档中的相应文档。

> **Tip**
>
> 最佳做法是使用系统分配的文档 `id` 作为外键。虽然您可以定义作为唯一标识符的字段（例如车辆的 VIN），但文档的真正唯一标识符是`id`。字段包含在文档元数据中，您可以在提交视图（系统定义的表格视图）中对其进行查询。
> 有关 QLDB 中的视图的更多信息，请参阅 核心概念。了解有关元数据的更多信息，请参阅 查询文档元数据。

设置示例数据

1. 审查以下 `.java` 文件。这些模型类代表您存储在 `vehicle-registration` 表格中的文档。它们可在 Amazon Ion 格式之间进行序列化。

   > **Note**
   >
   > Amazon QLDB 文档以 Ion 格式 (JSON 父集) 存储。因此，您可使用 FasterXML Jackson 库对 JSON 中的数据进行建模。

   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. 查看以下文件 (SampleData.java)，该文件代表您插入 vehicle-registration 表中的示例数据。

   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**
>
> 对于 Amazon Ion 软件包，您必须在应用程序中使用命名空间`com.amazon.ion`。AWS SDK for Java依赖于`software.amazon.ion`命名空间下的另一个 Ion 包，但这是一个与 QLDB 驱动程序不兼容的旧包。

```
/*
 * 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**
>
> - 该类使用 Ion 库提供辅助方法，用于将您的数据与 Ion 格式进行相互转换。
> - 该 `getDocumentId`方法对带有前缀`_ql_committed_`的表运行查询。这是保留的前缀，表示您要查询表的已提交视图。在此视图中，您的数据嵌套至`data`字段中，元数据嵌套至`metadata`字段中。

3. 编译并运行以下程序 (`CreateTable.java`) 以创建前面提到的表。

   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**
>
> 该程序演示了如何将 `TransactionExecutor` lambda 传递至`execute`方法。在此示例中，您使用 lambda 表达式在单个事务中运行多个`CREATE TABLE` PartiQL 语句。
> 该 `execute` 方法采用隐式启动事务，运行 lambda 中的所有语句，然后自动提交事务。

4. 如前所述，编译并运行以下程序 (`CreateIndex.java`)，以在表上创建索引。

   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. 编译并运行以下程序 (`InsertDocument.java`)，将示例数据插入表内。

   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 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**
>
> - 该程序演示如何使用参数化值调用`execute`方法。除了要运行的 PartiQL 语句之外，您还可以传递`IonValue`类数据参数。在语句字符串中将问号 (?) 作为变量占位符。
> - 如果 INSERT 语句成功，则返回每个插入文档的`id`。

接下来，您可以使用 SELECT 语句从 vehicle-registration 分类账中的表中读取数据。继续执行步骤 4：查询分类账中的表。

步骤 4：查询分类账中的表

在 Amazon QLDB 分类账中创建表格和加载数据后，您可运行查询以查看刚刚插入的车辆登记数据。QLDB 使用 PartiQL作为其查询语言，使用 Amazon Ion作为其面向文档的数据模型。

PartiQL 是开源、与 SQL 兼容的查询语言，现已扩展为可与 Ion 配合使用。使用 PartiQL，您可使用熟悉的 SQL 运算符插入、查询和管理数据。Amazon Ion 是 JSON 的超集。Ion 是基于文档的开源数据格式，可让您灵活地存储和处理结构化、半结构化和嵌套数据。

在此步骤中，您将使用SELECT 语句从 vehicle-registration分类账中的表中读取数据。

> **Warning**
>
> 当您在没有索引查找的情况下在 QLDB 中运行查询时，它会调用全表扫描。PartiQL 之所以支持此类查询，是因为其与 SQL 兼容。但是，切勿在 QLDB 中对生产用例运行表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
>
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (WHERE indexedField = 123或WHERE indexedField IN (456, 789)) 运行带有WHERE谓词子句的语句。有关更多信息，请参阅优化查询性能。

查询表格

- 编译并运行以下程序 (FindVehicles.java)，以查询分类账中某人名下注册的所有车辆。

  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**
>
> 首先，该程序在 Person 表格中查询文档GovId LEWISR261LL，以获取其id元数据字段。
> 然后，它使用文档id，通过PrimaryOwner.PersonId作为外键查询VehicleRegistration表。它还结合VIN字段上的Vehicle表VehicleRegistration。

要了解如何修改 vehicle-registration 分类账表格中的文档，请参阅 步骤 5：修改分类账中的文档。

步骤 5：修改分类账中的文档

现在您有了要处理的数据，可以开始在 Amazon QLDB 中对vehicle-registration分类账文档进行更改。在此步骤中，以下代码示例介绍了如何运行数据操作语言 (DML) 语句。这些声明更新一辆车的主要车主，并为另一辆车添加了次要车主。

修改文档

1. 编译并运行以下程序 (TransferVehicleOwnership.java)，使用分类账中的 VIN1N4AL11D75C109151 更新车辆的主要车主。

   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. 编译并运行以下程序 (AddSecondaryOwner.java)，将二级车主添加至分类账中带有 VIN KM8SRDHF6EU074761 的车辆中。

   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.");
    }
}
```

若要 查看 vehicle-registration 分类账中的这些更改，请参阅步骤 6：查看文档修订历史记录。

步骤 6：查看文档修订历史记录

在上一步中修改车辆注册数据后，您可查询其所有注册车主的历史记录以及任何其他更新的字段。在此步骤中，您将在vehicle-registration分类账的 VehicleRegistration表格中查询文档的修订历史记录。

若要查看修订历史记录

1. 查看以下程序 (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**
>
> - 您可以通过以下语法查询内置版本，以查看文档历史记录函数的修订历史记录。
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - 开始时间和结束时间均为可选。它们是 Amazon Ion 的字面值，可以用反引号 (`...`) 表示。要了解更多信息，请参阅 在 Amazon QLDB 中使用 PartiQL 查询 Ion。
> - 最佳做法是，使用日期范围（开始时间 和 结束时间）和文档 ID (`metadata.id`) 来限定历史查询。QLDB 处理事务中的`SELECT`查询，这些查询受事务超时限制的约束。
>
>   QLDB 历史记录按文档 ID 编制索引，目前无法创建其他历史索引。包含开始时间和结束时间的历史记录查询将从日期范围限定中获得便利。

2. 编译并运行 `QueryHistory.java` 程序，以使用 VIN `1N4AL11D75C109151`查询`VehicleRegistration`文档的修订历史记录。

要以加密方式验证 `vehicle-registration` 分类账中的文档修订版，请继续第 7 步：验证分类账中的文档。

第 7 步：验证分类账中的文档

借助 Amazon QLDB，您可在 SHA-256 中使用加密哈希，从而有效地验证分类账日记账中文档的完整性。要详细了解验证和加密哈希在 QLDB 中的工作原理，请参阅 Amazon QLDB 中的数据验证。

在此步骤中，您将验证`vehicle-registration`分类账`VehicleRegistration`表格中的单据修订版本。首先，您请求一份摘要，该摘要作为输出文件返回，并作为分类账整个变更历史记录的签名。然后，您要求提供与此摘要相关的修订证明。使用此证明，如果所有验证检查都可通过，可以验证修订版的完整性。

验证文档的修订版

1. 查看以下 `.java` 文件，这些文件代表了验证所需 QLDB 对象，以及带有 Ion 和字符串值的辅助方法的实用程序类。

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. 使用两个.java文件 (GetDigest.java和GetRevision.java) 执行以下步骤：

   - 从vehicle-registration分类账中请求新摘要。
   - 请索取 VehicleRegistration 表格中每份文件修订版的证明。
   - 通过重新计算摘要，使用返回的摘要和证明验证修订版。

   GetDigest.java 程序包含以下代码。

```
/*
 * 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**
>
> 使用该 `getDigest` 方法请求一份涵盖分类账中日记账当前提示的摘要。日记账提示指的是 QLDB 收到您的请求时的最近提交的数据块。

`GetRevision.java` 程序包含以下代码。

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**
>
> 在getRevision方法返回指定文档修订版本的证明后，此程序使用客户端 API 验证该修订版。有关此 API 使用的算法的概述，请参阅 使用证明重新计算您的摘要。

3. 编译并运行GetRevision.java程序，使用 VIN 1N4AL11D75C109151 对 VehicleRegistration 文档进行加密验证。

要导出和验证 vehicle-registration 分类账中的日记账数据，请继续 步骤 8：导出并验证分类账中的日记账数据。

步骤 8：导出并验证分类账中的日记账数据

在 Amazon QLDB 中，您可出于各种目的访问分类账中的日记账内容，例如数据保留、分析和审计。有关更多信息，请参阅从Amazon QLDB 导出日记账数据。

在此步骤中，您将`vehicle-registration`分类账中的日记账数据块导出至 Amazon S3 存储桶中。然后，您可使用导出的数据验证日记账数据块和每个区块中各个哈希组件之间的哈希链。

您使用 AWS Identity and Access Management (IAM) 委托人实体必须具有足够的 IAM 权限才能在您的 AWS 账户中创建 Amazon S3 存储桶。有关更多信息，请参阅 Amazon S3 用户指南 中的 Policies and Permissions in Amazon S3 中的策略和权限。您还必须有权创建附带权限策略的 IAM 角色，该策略允许 QLDB 将对象写入至您的 Amazon S3 存储桶。有关更多信息，请参阅 IAM 用户指南中的访问 IAM 资源所需的权限。

导出和验证日记账数据

1. 查看以下文件 (`JournalBlock.java`)，该文件代表日记账数据块及其数据内容。它包括一个名为`verifyBlockHash()`的方法，该方法演示了如何计算区块哈希的每个组成部分。

```
/*
 * 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. 编译并运行以下程序 (ValidateQldbHashChain.java)，以执行以下步骤：

   1. 将vehicle-registration分类账中的日记账数据块导出到名为**qldb-tutorial-journal-export-*111122223333***（用您的AWS 账户数字替换）的 Amazon S3 存储桶。
   2. 通过调用verifyBlockHash()来验证每个区块中的各个哈希组件。
   3. 验证日记账数据块之间的哈希链。

```
/*
 * 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;
        }
    }

}
```

如果您不再需要使用vehicle-registration分类账，请继续步骤 9：（可选）清除资源。

步骤 9：（可选）清除资源

您可以继续使用 vehicle-registration 分类账。但是，如果您不再需要，请将其删除。

删除分类账

1. 编译并运行以下程序 (DeleteLedger.java) 以删除您的 vehicle-registration 分类账及其所有内容。

```
/*
 * 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**
>
> 如果您的分类账启用了删除保护，您必须先禁用它，然后才能使用 QLDB API 删除分类账。

2. 如果您在上一步中导出了日记账数据，但不再需要这些数据，请使用 Amazon S3 控制台删除 S3 存储桶。

   通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

Amazon QLDB Node.js 教程

在本教程样例应用程序的实现中，您将使用 Amazon QLDB 驱动程序通过适用于 Node.js 中的 JavaScript 的 AWS SDK 来创建 QLDB 分类账，并用样例数据填充它。

在学习本教程时，您可以参考 AWS SDK for JavaScript API 参考 以了解管理 API 操作。有关事务数据操作，您可参见适用于 Node.js API 参考的 QLDB 驱动程序。

> **Note**
>
> 在适用的情况下，某些用例对于 Node.js 的 QLDB 驱动程序的每个支持主要版本都配备不同的命令或代码示例。

主题

安装 Amazon QLDB Node.js 示例应用程序

本节介绍如何按照分步展示的 Node.js 教程，安装和运行 Amazon QLDB 示例应用程序。此示例应用程序的用例是机动车辆部门 (DMV) 数据库，用于追踪有关车辆登记的完整历史信息。

适用于 Node.js 的 DMV 示例应用程序是 GitHub 存储库中的开放资源，存储库网址为aws-samples/amazon-qldb-dmv-sample-python。

先决条件

在开始之前，请确保您已完成适用于 Node.js 先决条件 的 QLDB 驱动程序。这包括安装 Node.js 并执行以下操作：

1. 注册AWS。
2. 创建具有适当 QLDB 权限的用户。
3. 授权以编程方式访问开发。

要完成本教程中的所有步骤，您需要通过 QLDB API 对分类账资源拥有完全管理权限。

安装

要安装示例应用程序

1. 输入以下命令，以从 GitHub 克隆示例应用程序。

   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
   ```

   示例应用程序打包了本教程中的完整源代码及其依赖项，包括 Node.js 驱动程序和 AWSNode.js 中适用于 JavaScript 的 SDK。这个应用程序是用 TypeScript 编写的。

2. 切换到克隆 `amazon-qldb-dmv-sample-nodejs` 包的目录。

```
cd amazon-qldb-dmv-sample-nodejs
```

3. 对依赖项进行全新安装。

```
npm ci
```

4. 转换包。

```
npm run build
```

转换后的 JavaScript 文件写在 `./dist` 目录中。

5. 继续 第 1 步：创建新分类账 开始教程并创建分类账。

第 1 步：创建新分类账

在此步骤中，您将创建一个名为 `vehicle-registration` 的新 Amazon QLDB 分类账。

创建新的分类账

1. 查看以下文件 (`Constants.ts`)，其包含本教程中所有其他程序使用的常量值。

```
/*
 * 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. 使用以下程序 (`CreateLedger.ts`) 创建名为 `vehicle-registration` 的分类账。

```
/*
 * 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**
>
> - 在 `createLedger` 调用中，您必须指定分类账名称和权限模式。我们建议使用 STANDARD 权限模式来最大限度地提高分类账数据的安全性。
> - 创建分类账时，将默认启用 删除保护。QLDB 中有一项功能，可防止分类账被任何用户删除。您可以选择使用 QLDB API 或 AWS Command Line Interface (AWS CLI) 在创建分类账时禁用删除保护。
> - 您还可选择指定要附加到分类账的标签。

3. 要运行编译后的程序，请输入以下命令。

```
node dist/CreateLedger.js
```

要验证您与新分类账的连接，请继续 步骤 2：测试分类账的连接性。

步骤 2：测试分类账的连接性

在此步骤中，您将验证是否可以使用事务数据 API 端点连接至 Amazon QLDB 中的`vehicle-registration`分类账。

测试与分类账的连接性

1. 使用以下程序 (`ConnectToLedger.ts`)，该程序创建了与`vehicle-registration` 分类账的数据会话连接。

   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**
>
> 要在分类账上运行数据事务，必须创建一个 QLDB 驱动对象以连接到指定的分类账。这与您在 上一步 中创建分类账时使用的 `qldbClient` 客户端对象不同。之前的客户端仅用于运行Amazon QLDB API 参考中列出的管理 API 操作。

2. 要运行编译后的程序，请输入以下命令。

```
node dist/ConnectToLedger.js
```

要在 `vehicle-registration` 分类账中创建表，请继续 第 3 步：创建表、索引与示例数据。

第 3 步：创建表、索引与示例数据

当您的 Amazon QLDB 分类账处于活动状态且接受连接时，您可开始创建有关车辆、车主和注册信息的数据表。创建表和索引后，可以向其中加载数据。

在此步骤中，您将在`vehicle-registration`分类账中创建四个表格：

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

您还可创建以下索引。

表名称	字段
VehicleRegistration	VIN
VehicleRegistration	LicensePlateNumber
Vehicle	VIN
Person	GovId
DriversLicense	LicenseNumber
DriversLicense	PersonId

插入示例数据时，首先要在 Person 表格中插入文档。然后使用系统分配的、来自每个Person文档的id填充适当VehicleRegistration 和 DriversLicense文档中的相应文档。

> **Tip**
>
> 最佳做法是使用系统分配的文档 id 作为外键。虽然您可以定义作为唯一标识符的字段（例如车辆的 VIN），但文档的真正唯一标识符是id。字段包含在文档元数据中，您可以在提交视图（系统定义的表格视图）中对其进行查询。
> 有关 QLDB 中的视图的更多信息，请参阅核心概念。了解有关元数据的更多信息，请参阅 查询文档元数据。

创建表和索引

1. 使用以下程序 (CreateTable.ts) 以创建前面提到的表。

```
/*
 * 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**
>
> 该程序演示如何在 QLDB 驱动程序实例中使用 `executeLambda` 函数。在此示例中，您使用单个 lambda 表达式运行多个CREATE TABLE PartiQL 语句。
> 该执行函数采用隐式启动事务，运行 lambda 中的所有语句，然后自动提交事务。

2. 要运行编译后的程序，请输入以下命令。

```
node dist/CreateTable.js
```

3. 如前所述，使用以下程序 (`CreateIndex.ts`)，以在表上创建索引。

```
/*
 * 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. 要运行编译后的程序，请输入以下命令。

```
node dist/CreateIndex.js
```

将样本数据加载到表中

1. 审查以下 .ts 文件。

   1. SampleData.ts - 包含您插入到 vehicle-registration 表中的示例数据。

```
/*
 * 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 - 从 ion-js 包导入的实用程序模块，提供用于转换、解析和打印 Amazon Ion 数据的辅助函数。

```
/*
 * 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**
>
> getDocumentId 函数运行一个查询，从表中返回系统分配的文档 ID。要了解更多信息，请参阅 通过 BY 子句查询文档 ID。

2. 编译并运行以下程序 (InsertDocument.ts)，将示例数据插入表内。

```
/*
 * 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**
>
> - 该程序演示如何使用参数化值调用`execute`方法。除了要运行的 PartiQL 语句之外，您还可以传递类数据参数。在语句字符串中将问号 (?) 作为变量占位符。
> - 如果 INSERT 语句成功，则返回每个插入文档的`id`。

3. 要运行编译后的程序，请输入以下命令。

```
node dist/InsertDocument.js
```

接下来，您可以使用 SELECT 语句从 vehicle-registration 分类账中的表中读取数据。继续执行步骤 4：查询分类账中的表。

步骤 4：查询分类账中的表

在 Amazon QLDB 分类账中创建表格和加载数据后，您可运行查询以查看刚刚插入的车辆登记数据。QLDB 使用 PartiQL作为其查询语言，使用 Amazon Ion作为其面向文档的数据模型。

PartiQL 是开源、与 SQL 兼容的查询语言，现已扩展为可与 Ion 配合使用。使用 PartiQL，您可使用熟悉的 SQL 运算符插入、查询和管理数据。Amazon Ion 是 JSON 的超集。Ion 是基于文档的开源数据格式，可让您灵活地存储和处理结构化、半结构化和嵌套数据。

在此步骤中，您将使用 SELECT 语句从 vehicle-registration分类账中的表中读取数据。

> **Warning**
>
> 当你在没有索引查找的情况下运行查询时，它会调用全表扫描。PartiQL 之所以支持此类查询，是因为其与 SQL 兼容。但是，切勿在 QLDB 中对生产用例运行表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (WHERE indexedField = 123或WHERE indexedField IN (456, 789)) 运行带有WHERE谓词子句的语句。有关更多信息，请参阅优化查询性能。

查询表格

1. 编译并运行以下程序 (FindVehicles.ts)，以查询分类账中某人名下注册的所有车辆。

```
/*
 * 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**
>
> 首先，该程序在 Person 表格中查询文档GovId LEWISR261LL，以获取其id元数据字段。
> 然后，它使用文档id，通过PrimaryOwner.PersonId作为外键查询VehicleRegistration表。它还结合VIN字段上的Vehicle表的VehicleRegistration。

2. 要运行编译后的程序，请输入以下命令。

```
node dist/FindVehicles.js
```

要了解如何修改 vehicle-registration 分类账表格中的文档，请参阅 步骤 5：修改分类账中的文档。

步骤 5：修改分类账中的文档

现在您有了要处理的数据，可以开始在 Amazon QLDB 中对 vehicle-registration 分类账文档进行更改。在此步骤中，以下代码示例介绍了如何运行数据操作语言 (DML) 语句。这些声明更新一辆车的主要车主，并为另一辆车添加了次要车主。

修改文档

1. 使用以下程序 (TransferVehicleOwnership.ts)，更新分类账中带有 VIN 1N4AL11D75C109151 的车辆的主要车主。

```
/*
 * 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. 要运行编译后的程序，请输入以下命令。

```
node dist/TransferVehicleOwnership.js
```

3. 使用以下程序 (AddSecondaryOwner.ts)，将二级车主添加至分类账中带有 VIN KM8SRDHF6EU074761 的车辆中。

```
/*
 * 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. 要运行编译后的程序，请输入以下命令。

```
node dist/AddSecondaryOwner.js
```

若要查看 vehicle-registration 分类账中的这些更改，请参阅步骤 6：查看文档修订历史记录。

步骤 6：查看文档修订历史记录

在上一步中修改车辆注册数据后，您可查询其所有注册车主的历史记录以及任何其他更新的字段。在此步骤中，您将在 vehicle-registration 分类账的 VehicleRegistration 表格中查询文档的修订历史记录。

若要查看修订历史记录

1. 编译并运行 QueryHistory.ts 程序，以使用 VIN 1N4AL11D75C109151查询VehicleRegistration文档的修订历史记录。

```
/*
 * 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

- 您可以通过以下语法查询内置版本，以查看文档历史记录函数的修订历史记录。

```
SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

- 开始时间和结束时间均为可选。它们是 Amazon Ion 的字面值，可以用反引号 (`...`) 表示。有关更多信息，请参阅在 Amazon QLDB 中使用 PartiQL 查询 Ion。

- 最佳做法是，使用日期范围（开始时间 和 结束时间）和文档 ID (`metadata.id`) 来限定历史记录查询。QLDB 处理事务中的`SELECT`查询，这些查询受事务超时限制的约束。

  QLDB 历史记录按文档 ID 编制索引，目前无法创建其他历史索引。包含开始时间和结束时间的历史记录查询将从日期范围限定中获得便利。

2. 要运行编译后的程序，请输入以下命令。

```
node dist/QueryHistory.js
```

要以加密方式验证 `vehicle-registration` 分类账中的文档修订版，请继续 第 7 步：验证分类账中的文档。

第 7 步：验证分类账中的文档

借助 Amazon QLDB，您可在 SHA-256 中使用加密哈希，从而有效地验证分类账日记账中文档的完整性。要详细了解验证和加密哈希在 QLDB 中的工作原理，请参阅 Amazon QLDB 中的数据验证。

在此步骤中，您将验证 `vehicle-registration` 分类账 `VehicleRegistration` 表格中的单据修订版本。首先，您请求一份摘要，该摘要作为输出文件返回，并作为分类账整个变更历史记录的签名。然后，您要求提供与此摘要相关的修订证明。使用此证明，如果所有验证检查都可通过，可以验证修订版的完整性。

验证文档的修订版

1. 查看以下 `.ts` 文件，其中包含验证所需的 QLDB 对象。

   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. 使用两个.ts程序 (GetDigest.ts和GetRevision.ts) 执行以下步骤：

   - 从 vehicle-registration 分类账中请求新摘要。
   - 请从VehicleRegistration 表格的 VIN 1N4AL11D75C109151 索取每份文件修订版的证明。
   - 通过重新计算摘要，使用返回的摘要和证明验证修订版。

   GetDigest.ts 文件包含以下代码。

```
/*
 * 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**
>
> 使用该 `getDigest` 方法请求一份涵盖分类账中日记账当前提示的摘要。日记账提示指的是 QLDB 收到您的请求时的最近提交的数据块。

`GetRevision.ts` 文件包含以下代码。

```
/*
 * 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**
>
> 在getRevision函数返回指定文档修订版本的证明后，此程序使用客户端 API 验证该修订版。

3. 要运行编译后的程序，请输入以下命令。

```
node dist/GetRevision.js
```

如果您不再需要使用vehicle-registration分类账，请继续第 8 步（可选）：清除资源。

第 8 步（可选）：清除资源

您可以继续使用 vehicle-registration 分类账。但是，如果您不再需要，请将其删除。

删除分类账

1. 使用以下程序 (DeleteLedger.ts) 以删除您的 vehicle-registration 分类账及其所有内容。

```
/*
 * 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**
>
> 如果分类账启用了删除保护，您必须先禁用它，然后才能使用 QLDB API 删除分类账。

2. 要运行编译后的程序，请输入以下命令。

```
node dist/DeleteLedger.js
```

Amazon QLDB Python 教程

在本教程样例应用程序的实现中，您将使用 Amazon QLDB 驱动程序通过 AWS SDK for Python (Boto3) 来创建 QLDB 分类账，并用样例数据填充它。

在学习本教程时，您可以参考 AWS 适用于 Python 的 QLDB Driver (Boto3) API 参考 中的 QLDB 低级客户端 来管理 API 操作。有关事务数据操作，您可参见适用于 Python 的 QLDB Driver API 参考。

> **Note**
>
> 在适用的情况下，某些用例对于 Python 的 QLDB 驱动程序的每个支持主要版本都配备不同的命令或代码示例。

主题

安装 Amazon QLDB Python 示例应用程序

本节介绍如何按照分步展示的 Python 教程，安装和运行 Amazon QLDB 示例应用程序。此示例应用程序的用例是机动车辆部门 (DMV) 数据库，用于追踪有关车辆登记的完整历史信息。

适用于 Python 的 DMV 示例应用程序是 GitHub 存储库中的开放资源，存储库网址为aws-samples/amazon-qldb-dmv-sample-python。

先决条件

在开始之前，请确保您已完成适用于 Python 先决条件 的 QLDB 驱动程序。这包括安装 Python 并执行以下操作：

1. 注册AWS。
2. 创建具有适当 QLDB 权限的用户。
3. 授权以编程方式访问开发。

要完成本教程中的所有步骤，您需要通过 QLDB API 对分类账资源拥有完全管理权限。

安装

要安装示例应用程序

1. 输入以下 `pip` 命令，以从 GitHub 克隆并安装示例应用程序。

   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
   ```

   示例应用程序打包了本教程中的完整源代码及其依赖项，包括 Python 驱动程序和 AWS SDK for Python (Boto3)。
2. 在命令行开始运行代码之前，请将当前工作目录切换到 `pyqldbsamples` 包的安装位置。输入以下命令。

   ```
   cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
   ```
3. 继续 第 1 步：创建新的分类账 开始教程并创建分类账。

第 1 步：创建新的分类账

在此步骤中，您将创建一个名为 `vehicle-registration` 的新 Amazon QLDB 分类账。

创建新的分类账

1. 查看以下文件 (constants.py)，其包含本教程中所有其他程序使用的常量值。

```
# 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. 使用以下程序 (create_ledger.py) 创建名为 vehicle-registration 的分类账。

```
# 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**
>
> - 在 `create_ledger` 调用中，您必须指定分类账名称和权限模式。我们建议使用 STANDARD 权限模式来最大限度地提高分类账数据的安全性。
> - 创建分类账时，将默认启用 删除保护。QLDB 中有一项功能，可防止分类账被任何用户删除。您可以选择使用 QLDB API 或 AWS Command Line Interface (AWS CLI) 在创建分类账时禁用删除保护。
> - 您还可选择指定要附加到分类账的标签。

3. 要运行该程序，请输入以下命令。

```
python create_ledger.py
```

要验证您与新分类账的连接，请继续 步骤 2：测试分类账的连接性。

步骤 2：测试分类账的连接性

在此步骤中，您将验证是否可以使用事务数据 API 端点连接至 Amazon QLDB 中的`vehicle-registration`分类账。

测试与分类账的连接性

1. 查看以下程序 (`connect_to_ledger.py`)，该程序创建了与 `vehicle-registration` 分类账的数据会话连接。

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**
>
> - 要在分类账上运行数据事务，必须创建一个 QLDB 驱动对象以连接到指定的分类账。这与您在上一步中创建分类账时使用的 `qldb_client` 客户端对象不同。之前的客户端仅用于Amazon QLDB API 参考中列出的管理 API 操作。
> - 创建此驱动程序对象时，您必须指定分类账名称。

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**
>
> - 要在分类账上运行数据事务，必须创建一个 QLDB 驱动程序对象以连接到指定的分类账。这与您在上一步中创建分类账时使用的 `qldb_client` 客户端对象不同。此前的客户端仅用于Amazon QLDB API 参考中列出的管理 API 操作。
> - 首先，创建一个池化的 QLDB 驱动程序对象。创建此驱动程序时，您必须指定分类账名称。
> - 然后，您可以从此池驱动程序对象创建会话。

2. 要运行该程序，请输入以下命令。

```
python connect_to_ledger.py
```

要在 `vehicle-registration` 分类账中创建表，请继续 步骤 3：创建表、索引与示例数据。

步骤 3：创建表、索引与示例数据

当您的 Amazon QLDB 分类账处于活动状态且接受连接时，您可开始创建有关车辆、车主和注册信息的数据表。创建表和索引后，可以向其中加载数据。

在此步骤中，您将在`vehicle-registration`分类账中创建四个表格：

- VehicleRegistration
- Vehicle
- Person
- DriversLicense

您还可创建以下索引。

表名称	Field
VehicleRegistration	VIN
VehicleRegistration	LicensePlateNumber
Vehicle	VIN
Person	GovId
DriversLicense	LicenseNumber
DriversLicense	PersonId

插入示例数据时，首先要在 Person 表格中插入文档。然后使用系统分配的、来自每个Person文档的id填充适当VehicleRegistration 和 DriversLicense文档中的相应文档。

> **Tip**
>
> 最佳做法是使用系统分配的文档 id 作为外键。虽然您可以定义作为唯一标识符的字段（例如车辆的 VIN），但文档的真正唯一标识符是id。字段包含在文档元数据中，您可以在提交视图（系统定义的表格视图）中对其进行查询。
> 有关 QLDB 中的视图的更多信息，请参阅核心概念。了解有关元数据的更多信息，请参阅 查询文档元数据。

创建表和索引

1. 编译并运行以下程序 (create_table.py) 以创建前面提到的表。

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**
>
> 该程序演示如何使用 `execute_lambda` 函数。在此示例中，您使用 lambda 表达式在单个事务中运行多个CREATE TABLE PartiQL 语句
> 该 方法采用隐式启动事务，运行 lambda 中的所有语句，然后自动提交事务。

2. 要运行该程序，请输入以下命令。

```
python create_table.py
```

3. 如前所述，编译并运行以下程序 (`create_index.py`)，以在表上创建索引。

   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. 要运行该程序，请输入以下命令。

```
python create_index.py
```

将样本数据加载到 表中

1. 查看以下文件 (sample_data.py)，该文件代表您插入 vehicle-registration 表中的示例数据。此文件也从 amazon.ion 软件包导出，以提供用于转换、解析和打印 Amazon Ion 数据的辅助函数。

```
# 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**
>
> `get_document_ids` 函数运行一个查询，从表中返回系统分配的文档 ID。要了解更多信息，请参阅 通过 BY 子句查询文档 ID。

2. 编译并运行以下程序 (`insert_document.py`)，将示例数据插入表内。

   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**
>
> - 该程序演示如何使用参数化值调用`execute_statement`方法。除了要运行的 PartiQL 语句之外，您还可以传递类数据参数。在语句字符串中将问号 (?) 作为变量占位符。
> - 如果 INSERT 语句成功，则返回每个插入文档的`id`。

3. 要运行该程序，请输入以下命令。

```
python insert_document.py
```

接下来，您可以使用 SELECT 语句从 `vehicle-registration` 分类账中的表中读取数据。继续执行步骤 4：查询分类账中的表。

步骤 4：查询分类账中的表

在 Amazon QLDB 分类账中创建表格和加载数据后，您可运行查询以查看刚刚插入的车辆登记数据。QLDB 使用 PartiQL作为其查询语言，使用 Amazon Ion作为其面向文档的数据模型。

PartiQL 是开源、与 SQL 兼容的查询语言，现已扩展为可与 Ion 配合使用。使用 PartiQL，您可使用熟悉的 SQL 运算符插入、查询和管理数据。Amazon Ion 是 JSON 的超集。Ion 是基于文档的开源数据格式，可让您灵活地存储和处理结构化、半结构化和嵌套数据。

在此步骤中，您将使用 SELECT 语句从 `vehicle-registration`分类账中的表中读取数据。

> **Warning**
>
> 当你在没有索引查找的情况下运行查询时，它会调用全表扫描。PartiQL 之所以支持此类查询，是因为其与 SQL 兼容。但是，切勿在 QLDB 中对生产用例运行表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。

为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`) 运行带有`WHERE`谓词子句的语句。有关更多信息，请参阅优化查询性能。

查询表格

1. 编译并运行以下程序 (`find_vehicles.py`)，以查询分类账中某人名下注册的所有车辆。

   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**
>
> 首先，该程序在 Person 表格中查询文档GovId LEWISR261LL，以获取其id元数据字段。

然后，它使用文档id，通过PrimaryOwner.PersonId作为外键查询VehicleRegistration表。它还结合VIN字段上的Vehicle表VehicleRegistration。

2. 要运行该程序，请输入以下命令。

```
python find_vehicles.py
```

要了解如何修改 vehicle-registration 分类账表格中的文档，请参阅 步骤 5：修改分类账中的文档。

步骤 5：修改分类账中的文档

现在您有了要处理的数据，可以开始在 Amazon QLDB 中对 vehicle-registration 分类账文档进行更改。在此步骤中，以下代码示例介绍了如何运行数据操作语言 (DML) 语句。这些声明更新一辆车的主要车主，并为另一辆车添加了次要车主。

修改文档

1. 使用以下程序 (transfer_vehicle_ownership.py)，更新分类账中带有 VIN 1N4AL11D75C109151 的车辆的主要车主。

   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. 要运行该程序，请输入以下命令。

```
python transfer_vehicle_ownership.py
```

3. 使用以下程序 (add_secondary_owner.py)，将二级车主添加至分类账中带有 VIN KM8SRDHF6EU074761 的车辆中。

   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. 要运行该程序，请输入以下命令。

```
python add_secondary_owner.py
```

若要查看 vehicle-registration 分类账中的这些更改，请参阅步骤 6：查看文档修订历史记录。

步骤 6：查看文档修订历史记录

在上一步中修改车辆注册数据后，您可查询其所有注册车主的历史记录以及任何其他更新的字段。在此步骤中，您将在 vehicle-registration 分类账的 VehicleRegistration 表格中查询文档的修订历史记录。

若要查看修订历史记录

1. 编译并运行 query_history.py 程序，以使用 VIN 1N4AL11D75C109151查询VehicleRegistration文档的修订历史记录。

   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**
>
> - 您可以通过以下语法查询内置版本，以查看文档历史记录函数的修订历史记录。
>
>   ```
>   SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
>   [ WHERE h.metadata.id = 'id' ]
>   ```
>
> - 开始时间和结束时间均为可选。它们是 Amazon Ion 的字面值，可以用反引号 (`...`) 表示。要了解更多信息，请参阅 在 Amazon QLDB 中使用 PartiQL 查询 Ion。
> - 最佳做法是，使用日期范围（开始时间 和 结束时间）和文档 ID (`metadata.id`) 来限定历史记录查询。QLDB 处理事务中的`SELECT`查询，这些查询受事务超时限制的约束。
>
>   QLDB 历史记录按文档 ID 编制索引，目前无法创建其他历史索引。包含开始时间和结束时间的历史记录查询将从日期范围限定中获得便利。

2. 要运行该程序，请输入以下命令。

```
python query_history.py
```

要以加密方式验证 `vehicle-registration` 分类账中的文档修订版，请继续 第 7 步：验证分类账中的文档。

第 7 步：验证分类账中的文档

借助 Amazon QLDB，您可在 SHA-256 中使用加密哈希，从而有效地验证分类账日记账中文档的完整性。要详细了解验证和加密哈希在 QLDB 中的工作原理，请参阅 Amazon QLDB 中的数据验证。

在此步骤中，您将验证 `vehicle-registration` 分类账 `VehicleRegistration` 表格中的单据修订版本。首先，您请求一份摘要，该摘要作为输出文件返回，并作为分类账整个变更历史记录的签名。然后，您要求提供与此摘要相关的修订证明。使用此证明，如果所有验证检查都可通过，可以验证修订版的完整性。

验证文档的修订版

1. 查看以下 .py 文件，这些文件代表了验证所需的 QLDB 对象，以及一个带有用于将 QLDB 响应类型转换为字符串的辅助函数的实用模块。

   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. 使用两个.py文件 (get_digest.py和get_revision.py) 执行以下步骤：

   - 从 vehicle-registration 分类账中请求新摘要。
   - 请从VehicleRegistration 表格的 VIN 1N4AL11D75C109151 索取每份文件修订版的证明。
   - 通过重新计算摘要，使用返回的摘要和证明验证修订版。

   get_digest.py 文件包含以下代码。

```
# 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**
>
> 使用该 `get_digest_result` 方法请求一份涵盖分类账中日记账当前提示的摘要。日记账提示指的是 QLDB 收到您的请求时的最近提交的数据块。

`get_revision.py` 文件包含以下代码。

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**
>
> 在get_revision方法返回指定文档修订版本的证明后，此程序使用客户端 API 验证该修订版。

3. 要运行该程序，请输入以下命令。

```
python get_revision.py
```

如果您不再需要使用vehicle-registration分类账，请继续第 8 步（可选）：清除资源。

第 8 步（可选）：清除资源

您可以继续使用 vehicle-registration 分类账。但是，如果您不再需要，请将其删除。

删除分类账

1. 使用以下程序 (delete_ledger.py) 以删除您的 vehicle-registration 分类账及其所有内容。

```
# 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**
>
> 如果分类账启用了删除保护，您必须先禁用它，然后才能使用 QLDB API 删除分类账。

delete_ledger.py 文件还依赖于以下程序 (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. 要运行该程序，请输入以下命令。

```
python delete_ledger.py
```

使用 Amazon QLDB 中的 Amazon Ion 数据类型

Amazon QLDB 以 Amazon Ion格式存储数据。要在 QLDB 中处理数据，必须使用 Ion 库作为支持的编程语言依赖项。

本节学习如何将数据从原生类型转换为离子等效物，反之亦然。本参考指南显示了使用 QLDB 驱动程序处理 QLDB 分类账 Ion 数据代码示例。它包括 Java、.NET (C#)、Go、Node.js (TypeScript) 和 Python 代码示例。

主题

先决条件

以下代码示例假设您有一个 QLDB 驱动程序实例，该实例连接到一个名为`ExampleTable`的表的活动分类账。该表文档中，有一个文档包含以下八个字段：

- ExampleBool
- ExampleInt
- ExampleFloat

- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> 就本参考而言，假定存储在每个字段中的类型与其名称相匹配。实际上，QLDB 并不强制文档字段架构或数据类型定义。

Bool

以下代码示例介绍如何处理 Ion 布尔类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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

以下代码示例介绍如何处理 Ion 整数类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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

以下代码示例介绍如何处理 Ion 浮动类型。

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

使用 C# 浮点数

```
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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为IResult。

使用 C# 双

```
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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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))
```

十进制

以下代码示例介绍如何处理 Ion 小数类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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**
>
> 您还可以使用`ionValue.numberValue()`将 Ion 十进制数转换为 JavaScript 数字。使用`ionValue.numberValue()`具有更好的性能，但精度较低。

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))
```

时间戳

以下代码示例介绍如何处理 Ion 时间戳类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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))
```

字符串

以下代码示例介绍如何处理 Ion 字符串类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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

以下代码示例介绍如何处理 Ion 二进制大对象类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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))
```

List

以下代码示例介绍如何处理 Ion 列表类型。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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

在 QLDB 中，`struct`数据类型与其他 Ion 类型特别不同。插入到表格中的顶级文档必须属于`struct`类型。文档字段也可以存储嵌套`struct`。

为简单起见，以下示例定义了一个仅包含 `ExampleString` 和 `ExampleInt` 字段的文档。

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;
});
```

或者，您可以使用 Jackson 库 将数据类型从 Ion 映射出入。该库支持其他 Ion 数据类型，但此示例重点介绍`struct`类型。

```
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

使用Amazon.QLDB.Driver.Serialization库将原生 C# 数据类型从 Ion 映射出入。该库支持其他 Ion 数据类型，但此示例重点介绍`struct`类型。

```
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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为IResult。

或者，你可以使用Amazon.IonDotnet.Builders库来处理 Ion 数据类型。

```
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))
```

空值与动态类型

QLDB 支持开放内容，且不强制文档字段定义架构或数据类型定义。您也可在 QLDB 文档中存储 Ion 空值。前述示例都假设返回的每种数据类型都是已知的，并且不是 null。以下示例说明如何在 Ion 数据类型未知或可能 null 时与其结合使用。

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

编组限制

在 Go 中，当你编组然后解组它们时，nil 值不会保留它们的类型。一个 nil 值编组为 Ion null，但是 Ion null 则解组为零值，而不是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!
```

向下转换至 JSON

如果您的应用程序需要 JSON 兼容性，则可以将 Amazon Ion 数据向下转换至 JSON。但是，在某些情况下，如果您的数据使用 JSON 中不存在的丰富 Ion 类型，则将 Ion 转换为 JSON 时会有损失。

有关 Ion 与 JSON 转换规则的详细信息，请参阅Amazon Ion Cookbook中的向下转换至 JSON。

在 Amazon QLDB 中处理数据与历史记录

以下主题提供了关于创建、读取、更新和删除 (CRUD) 语句的基本示例。您可使用QLDB 控制台或QLDB Shell上的PartiQL 编辑器手动运行这些语句。本指南还向您介绍以下流程：当您更改分类账时，QLDB 如何处理您的数据。

QLDB 支持PartiQL查询语言。

有关展示如何使用 QLDB 驱动程序以编程方式运行类似语句的代码示例，请参阅驱动程序入门中的教程。

> **Tip**
>
> 以下是在 QLDB 中使用 PartiQL 的提示和最佳实践小贴士：
>
> - 了解并发和事务限制 — 查询SELECT等所有语句都应遵守乐观并发控制 (OCC) 冲突和事务限制，包括 30 秒事务暂停。
> - 使用索引 - 使用高基数索引，并运行有针对性的查询来优化语句并避免全表扫描。要了解更多信息，请参阅 优化查询性能。
> - 使用相等谓词 - 索引查找需要相等运算符 (= 或 IN)。不等式运算符 (<、>、LIKE、BETWEEN) 不符合索引查找的条件，因此会生成全表扫描。
> - 仅使用内部联接 - QLDB 仅支持内部联接。根据最佳实践标准，在为要加入的每个表编制索引的字段上进行联接。为联接条件与相等谓词选择高基数索引。

主题

- 查询系统目录
- 管理表
- 管理索引
- Amazon QLDB 中的唯一编号

创建带有索引的表和插入文档

创建 Amazon QLDB 分类账后，您的第一步是创建包含基本CREATE TABLE语句的表。表由QLDB 文档组成，这些版本是 Amazon Ion `struct` 格式的数据集。

主题

- 创建表与索引
- 插入文档

创建表与索引

表具有简单的、区分大小写的名称，没有名称空间。QLDB 支持开放内容且不强制架构，因此在创建表时不需要定义属性或数据类型。

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

CREATE TABLE语句返回系统为新表分配的 ID。QLDB 中的所有系统分配的 ID都是通用唯一标识符 (UUID)，每个标识符都以 Base62 编码的字符串表示。

> **Note**
>
> 或者，您可在创建表时为表资源定义标签。要了解如何操作，请参阅创建时对表格进行标记。

您还可以在表格创建索引以优化查询性能。

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

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

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

> **Important**
>
> QLDB 需要索引才能高效查找文档。如果没有索引，QLDB 在读取文档时需进行全表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (=或IN) 运行带有WHERE谓词子句的语句。有关更多信息，请参阅优化查询性能。

创建索引时应注意以下限制：

- 只能在单个顶级字段创建索引。不支持复合索引、嵌套索引、唯一索引以及基于函数的索引。
- 您可以为任何 Ion 数据类型创建索引，其中包括 `list` 和 `struct`。但是，无论 Ion 类型如何，您都只能通过整个 Ion 值进行索引查找。例如，使用 `list` 类型作为索引时，不能按列表中的一个项目进行索引查找。
- 只有使用相等谓词时，查询性能才会得到改善；例如`WHERE indexedField = 123` 或 `WHERE indexedField IN (456, 789)`。

  QLDB 不支持查询谓词不等式。因此，未实施范围过滤扫描。
- 索引字段名称区分大小写，且最大长度可为 128 个字符。
- 在 QLDB 中创建索引具有异步特点。非空表上完成索引所需的时间取决于表的大小。有关更多信息，请参阅管理索引。

插入文档

然后，您可将文档插入表格中。QLDB 文档以 Amazon Ion格式存储。以下 PartiQL INSERT语句包括Amazon QLDB 控制台入门中使用的车辆注册采样数据的子集。

```
INSERT INTO VehicleRegistration
<< {
    'VIN' : '1N4AL11D75C109151',
    'LicensePlateNumber' : 'LEWISR261LL',
    'State' : 'WA',
    'City' : 'Seattle',
    'PendingPenaltyTicketAmount' : 90.25,
    'ValidFromDate' : `2017-08-21T`,
    'ValidToDate' : `2020-05-11T`,
```

```
    'Owners' : {
        'PrimaryOwner' : { 'PersonId' : '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ { 'PersonId' : '5Ufgdlnj06gF5CWcOIu64s' } ]
    }
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'LicensePlateNumber' : 'CA762X',
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75,
    'ValidFromDate' : `2017-09-14T`,
    'ValidToDate' : `2020-06-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' },
        'SecondaryOwners' : []
    }
} >>
```

```
INSERT INTO Vehicle
<< {
    'VIN' : '1N4AL11D75C109151',
    'Type' : 'Sedan',
    'Year' : 2011,
    'Make' : 'Audi',
    'Model' : 'A5',
    'Color' : 'Silver'
} ,
{
    'VIN' : 'KM8SRDHF6EU074761',
    'Type' : 'Sedan',
    'Year' : 2015,
    'Make' : 'Tesla',
    'Model' : 'Model S',
    'Color' : 'Blue'
} >>
```

PartiQL 语法和语义

- 字段名称以单引号 (`'...'`) 括起来。
- 字符串值也以单引号 (`'...'`) 括起来。
- 时间戳用反引号 (`` `...` ``) 括起来。任何 Ion 字面值都可以用反引号表示。

- 整数和小数为不需要表示的字面值。

有关 PartiQL 的语法和语义详细信息，请参阅在 Amazon QLDB 中使用 PartiQL 查询 Ion。

INSERT 语句创建版本号为零的文档的初始修订版。为了唯一标识每个文档，QLDB 将分配文档 ID，以作为元数据的一部分。插入语句返回插入的每个文档 ID。

> **Important**
>
> 由于 QLDB 不强制执行架构，因此您可多次将同一个文档插入表中。每个插入语句向日志提交单独的文档条目，QLDB 为每个文档分配唯一的 ID。

若要了解如何查询您插入到表格中的文档，请继续 查询数据。

查询数据

用户视图 仅返回用户数据的最新未删除版本。这是 Amazon QLDB 中的默认视图。这意味着，当您只想查询自己的数据时，不需要特殊限定符。

有关以下查询示例的语法和参数的详细信息，请参阅Amazon QLDB PartiQL 参考中的SELECT。

主题

基本查询

基本 SELECT 查询会返回您插入表格中的文档。

> **Warning**
>
> 当你在没有索引查找的情况下运行查询时，它会调用全表扫描。PartiQL 之所以支持此类查询，是因为其与 SQL 兼容。但是，切勿在 QLDB 中对生产用例运行表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。

为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`) 运行带有`WHERE`谓词子句的语句。有关更多信息，请参阅优化查询性能。

以下查询显示了您之前在创建带有索引的表和插入文档中插入的车辆登记文件的结果。结果的顺序不指定，可能因每个`SELECT`查询而异。在 QLDB 中，任何查询都不应该依赖结果顺序。

```
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

在 PartiQL 中，您可使用单引号表示数据操作语言 (DML) 或查询语句中的字符串。但是 QLDB 控制台和 QLDB Shell 以 Amazon Ion 文本格式返回查询结果，因此您会看到用双引号括起来的字符串。

这种语法差异允许 PartiQL 查询语言保持 SQL 兼容性，并使 Amazon Ion 文本格式保持 JSON 兼容性。

投影和筛选

您可以进行预测 (定向SELECT) 和其他标准筛选条件 (WHERE条款)。以下查询返回 VehicleRegistration 表中文档字段的子集。它可筛选符合以下标准的车辆：

- 字符串筛选条件 — 它已在西雅图注册。
- 十进制筛选条件 — 它的待处理工单数量小于 100.0。
- 日期筛选退傲剑 — 其注册日期在 2019 年 9 月 4 日或之后有效。

```
SELECT r.VIN, r.PendingPenaltyTicketAmount, r.Owners
FROM VehicleRegistration AS r
WHERE r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
AND r.City = 'Seattle' --string
```

```
AND r.PendingPenaltyTicketAmount < 100.0 --decimal
AND r.ValidToDate >= `2019-09-04T` --timestamp with day precision
```

```
{
    VIN: "1N4AL11D75C109151",
    PendingPenaltyTicketAmount: 90.25,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

Joins

您也可编写内部联接查询。以下示例显示了隐式内部联接查询，该查询返回所有注册文件以及已注册车辆的属性。

```
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"
}
```

或者，您可使用如下显式语法编写相同的内部联接查询。

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

嵌套数据

您可在 QLDB 中使用 PartiQL 查询文档中的嵌套数据。以下示例展示了用于扁平化嵌套数据的相关子查询。此处的 @ 字符在技术上是可选的。但它明确表示你想在里面放置一个Owners结构VehicleRegistration，而不是一个名为不同的集合Owners（如有）。

```
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: []
}
```

下图展示了SELECT 列表中的子查询，除了内部联接外，该查询还投射嵌套数据。

```
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"]
}
```

以下查询返回表 VehicleRegistration 文档的 Owners.SecondaryOwners 列表中每个人的 PersonId 和索引（序数）。

```
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
}
```

要了解如何查询您的文档元数据，请继续 查询文档元数据。

查询文档元数据

INSERT 语句创建版本号为零的文档的初始修订版。为了唯一标识每个文档，Amazon QLDB 将分配文档 ID，以作为元数据的一部分。

除了文档 ID 和版本号外，QLDB 还会将系统为每个文档生成的其他元数据存储至表格。此元数据包含事务信息、日记账属性和文档的哈希值。

所有系统分配的 ID 都是通用唯一标识符 (UUID)，每个标识符都以 Base62 编码的字符串表示。有关更多信息，请参阅Amazon QLDB 中的唯一编号。

主题

已提交视图

您可以通过查询已提交视图访问文档元数据。此视图从系统定义的表返回文档，该表直接对应于您的用户表。它包括您的数据和系统生成的元数据的最新已提交、未删除修订版。要查询此视图，请在查询中的表名中添加前缀 `_ql_committed_`。(QLDB 中_ql_为系统对象保留了前缀。)

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

使用先前在创建带有索引的表和插入文档中插入的数据，此查询的输出显示了每个未删除文档的最新修订版的系统内容。系统文档的 `metadata` 字段中嵌套了元数据，您的用户数据嵌套在data字段中。

```
{
    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"
    }
}
```

已提交视图字段

- blockAddress — 分类账的日记账中提交文档修订的数据块的位置。可用于加密验证的地址包含以下两个字段。
  - strandId — 包含数据块的日记账链的唯一 ID。
  - sequenceNo — 一个索引号，用于指定数据块在链中的位置。

  > **Note**
  >
  > 本示例中的两个文档都具有相同的blockAddress，其sequenceNo相同。由于这些文档是在单个事务中插入的（在本例中为单个语句中），因此它们是在同一个数据块内提交的。

- hash — 唯一表示文档修订版本的 SHA-256 Ion 哈希值。哈希值涵盖了修订版 data 和 metadata 字段，可用于加密验证。
- data — 文档的用户数据属性。

  如果您对修订版进行编校，则此data结构将替换为dataHash字段，该字段的值是已删除data结构的 Ion 哈希。
- metadata — 文档的元数据属性。
  - id — 系统为文档分配的唯一 ID。
  - version — 文档的版本号。这是从零开始的整数，随着每个文档修订版本的增加而递增。
  - txTime – 文档修订提交到日记账的时间戳。
  - txId – 提交文档修订的事务唯一 ID。

加入已提交和用户视图

您可编写将已提交视图中的表与用户视图中的表连接起来的查询。例如，您可能希望将一个表id文档与另一个表的用户定义字段连接起来。

以下查询分别将两个名为 DriversLicense 和 Person的表与其 PersonId和的文档 id字段连接起来，后者使用已提交的视图。

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

要了解如何在默认用户视图中查询文档 ID 字段，请继续 通过 BY 子句查询文档 ID。

通过 BY 子句查询文档 ID

虽然您可以定义作为唯一标识符的字段（例如车辆的 VIN），但文档的真正唯一标识符是 id 元数据字段，如 插入文档 中所述。因此，您可以使用 id 字段在表之间创建关系。

文档 id 字段只能在已提交的视图中直接访问，但您也可使用BY子句将其投影到默认用户视图中。有关示例，请参阅以下查询和结果。

```
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" }]
    }
}
```

在此查询中，r_id是用户定义别名，在FROM子句中使用BY关键字声明。此 r_id 别名绑定至查询结果集中每个文档的 id 元数据字段。你可以在SELECT 子句中使用这个别名，也可以在用户视图的查询 WHERE 子句中使用这个别名。

但是，若要访问其他元数据属性，必须查询已提交视图。

使用证件 ID 加入

假设您正在使用一个表的id文档作为另一个表用户定义字段中的外键。您可以使用 BY 子句为这两个字段上的两个表编写内部联接查询(与上一主题中的加入已提交和用户视图类似)。

以下示例分别在它们的 PersonId 和文档 id 字段上连接两个名为 DriversLicense 和 Person 的表，对后者使用 BY 子句。

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
```

```
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

要了解如何对表格中的文档更改，请继续 更新和删除文档。

更新和删除文档

在 Amazon QLDB 中，文档修订版是一个 Amazon Ion 结构，它表示由唯一文档 ID 标识的文档序列的单个版本。每个修订版都包含文档完整数据集，包括您的用户数据和系统生成的元数据。每个文档修订版都由文档 ID 和从零开始的版本号组合作为唯一标识。

更新文档时，QLDB 会创建包含相同文档 ID 和递增版本号的新修订版。当从表格中删除文档时，文档的生命周期即告结束。这意味着不能再次创建包含相同文档 ID 的文档修订版。

对文档执行修订

例如，以下语句插入新车辆登记，更新登记城市，然后删除登记。这将导致对文档进行三次修订。

```
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**
>
> 插入语句和其他 DML 语句，会返回每个受影响文档的 ID。继续操作之前，请保存此 ID，因为下一个主题中的历史记录功能需要它。您还可以通过下述查询找到文档 OD。
>
> ```
> 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'
```

有关这些 DML 语句语法的更多示例和信息，请参阅Amazon QLDB PartiQL 参考中的更新 和 删除。

要在文档中插入和删除特定元素，可以使用以FROM关键字开头的UPDATE语句或者其他 DML 语句。有关更多信息和示例，请参阅 FROM（插入、删除或设置） 参考。

删除文档后，您将无法再在已提交视图或用户视图中对其进行查询。若要了解如何使用内置历史记录功能查询本文档的修订历史记录，请继续 查询修订历史记录。

查询修订历史记录

Amazon QLDB 将每个文档的完整历史记录存储至一个表格中。通过查询内置的历史记录功能，您可以查看之前在更新和删除文档中插入、更新和删除的车辆登记文件的所有三个修订版。

主题

- 历史记录函数
- 历史记录查询示例

历史记录函数

QLDB 中的历史函数是 PartiQL 扩展，它从表的系统定义视图返回修订版。因此，它将您的数据和关联元数据包含在与提交的视图相同的架构中。

语法

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Arguments

table_name | '***table_id***'

表名或表格 ID。表名是 PartiQL 标识符，可以用双引号或不用引号来表示。表 ID 必须用单引号将表文本括起来。要了解有关使用表 ID 的更多信息，请参阅 查询非活动表历史记录。

`***start-time***`，`***end-time***`

（可选）指定所有处于活动状态的修订时间范围。此参数并未指定在事务中向日记账提交修订的时间范围。

开始和结束时间是 Ion 时间戳文字，可以用反引号 (`` `...` ``) 表示。要了解更多信息，请参阅 在 Amazon QLDB 中使用 PartiQL 查询 Ion。

这些时间参数包含以下行为：

- 开始时间和结束时间均包含在内。它们必须采用ISO 8601日期和时间格式，并使用通用协调时间 (UTC)。
- 开始时间必须小于或等于结束时间，并且可以是过去任何任意日期。
- 结束时间必须小于或等于当前的 UTC 日期和时间。
- 如果您指定开始时间，但未指定结束时间，则查询会将结束时间默认为当前日期和时间。如果两者都未指定，则您的查询将返回整个历史记录。

'***id***'

（可选）若要查询其修订历史记录的文档 ID，用单引号表示。

> **Tip**
>
> 最佳做法是，使用日期范围（开始时间 和 结束时间）和文档 ID (`metadata.id`) 来限定历史记录查询。QLDB 中的每个 SELECT 查询都是在事务中处理的，并且受事务超时限制的约束。历史记录查询不使用您在表上创建的索引。QLDB 历史记录按文档 ID 编制索引，目前无法创建其他历史索引。包含开始时间和结束时间的历史记录查询将从日期范围限定中获得便利。

历史记录查询示例

要查询车辆登记文件的历史记录，请使用您之前在更新和删除文档中保存的`id`。例如，以下历史记录查询会返回在`2019-06-05T00:00:00Z`和`2019-06-05T23:59:59Z`之间处于活动状态的文档 ID `ADR2L11fGsU4Jr4EqTdnQF`的所有修订版。

> **Note**
>
> 切记，开始和结束时间参数并未指定在事务中向日记账提交修订的时间范围。例如，如果修订版本是在2019-06-05T00:00:00Z之前提交的，并且在该开始时间之后仍处于活动状态，则此示例查询将在结果中返回该修订版本。

请务必根据需要用自己的值替 换id、开始时间和结束时间。

```
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
```

您的查询结果应类似于以下内容。

```
{
    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"
    }
}
```

输出包含元数据属性，这些属性提供了有关每项何时修改以及由哪个事务修改的详细信息。从这些数据中，您可看到以下内容：

- 该文档其系统分配的id唯一标识：ADR2Ll1fGsU4Jr4EqTdnQF。这是以 Base62 编码的字符串表示的 UUID。
- INSERT 语句创建版本号为零的文档的初始修订版（版本 0）。
- 随后的每次更新都会创建一个具有相同文档 id 的新版本，并递增版本号。
- 该 txId 字段表示提交每个修订的事务，txTime显示每个修订的提交时间。
- DELETE 语句会创建文档的新修订版，也是最终修订版。此最终修订版仅仅包含元数据。

若要了解如何永久删除修订版，请继续 对文档修订版执行编校。

对文档修订版执行编校

在 Amazon QLDB，DELETE 语句只能通过创建将文档标记为已删除的新修订版，从逻辑上删除文档。QLDB 还支持数据编校操作，允许您永久删除表历史记录中的非活动文档修订版本。

> **Note**
>
> 2021 年 7 月 22 日之前创建的任何分类账目前都不符合编校资格。您可在 Amazon QLDB 控制台查看分类账的创建时间。

编校操作仅删除指定修订版中的用户数据，而日记账序列和文档元数据则保持不变。这样可保持分类账的整体数据完整性。

在开始在 QLDB 中进行数据编校之前，请务必在Amazon QLDB PartiQL 参考中查看修订注意事项和限制。

主题

- 编校存储过程
- 检查编校是否已完成
- 编校示例
- 删除和编校当前修订版本
- 编校修订版本中的特定字段

编校存储过程

您可以使用 REDACT_REVISION 存储过程永久删除分类账中单个非活动修订版本。此存储过程将删除索引存储和日志存储中指定修订版本中的所有用户数据。但是，它将使日记账序列和文档元数据 (包括文档 ID 和哈希) 保持不变。该操作不可逆。

指定的文档修订版本必须是在历史记录中处于非活动状态的修订版。文档的最新有效修订版不符合此编校条件。

若要编校多个修订版，必须为每个修订版运行一次存储进程。您可为每笔事务编校一个修订版。

语法

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Arguments

`*block-address*`

要编校的文档修订版本的日记账数据块位置。地址是一种包含两个字段的 `strandId`Amazon Ion 结构：即 和 `sequenceNo`。

这是一个 Ion 字面值，以反引号表示。例如：

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'*table-id*'

待编校修订版本的表的唯一 ID，用单引号表示。

'*document-id*'

待编校修订版本的唯一文档 ID，用单引号表示。

检查编校是否已完成

通过运行此存储过程提交编校请求后，QLDB 将异步处理数据的编校。完成后，修订版中的用户数据 (由 `data` 结构表示) 将被永久删除。若要检查编校请求是否已完成，您可以使用以下方法之一：

- 日记账导出
- 日记账流

- GetBlock API 操作
- GetRevision API 操作
- 历史记录函数— 注意： 在日记账中完成编校后，历史记录查询可能需要一段时间才能显示编校结果。异步编校完成后，您可能会看到一些修订版本在其他修订版本之前被编校，但是历史记录查询最终会显示已完成结果。

修订版本编校完成后，修订版本的 data 结构将被新dataHash字段所取代。该字段的值是已移除 data 结构的 Ion 哈希，如下例所示。因此，分类账保持了其整体数据的完整性，并通过现有验证 API 操作保持加密可验证性。要了解有关验证的更多信息，请参阅Amazon QLDB 中的数据验证。

编校示例

以您之前在查询修订历史记录中查看过的车辆登记文件为例。假设您要编校第二个修订版 (version:1)。以下查询示例显示了编校之前的此修订版。在查询结果中，将要编校的data结构以#### 突出显示。

```
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"
    }
}
```

请注意查询结果中的blockAddress，因为您需要将此值传递给REDACT_REVISION存储过程。然后，通过查询系统目录找到VehicleRegistration 表的唯一 ID，如下所示。

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

使用此表 ID 以及文档 ID 和数据块地址即可运行REDACT_REVISION。表 ID 和文档 ID 是字符串文字，必须用单引号括起来，数据块地址是用反引号括起的 Ion 文字。确保将这些参数替换为自己的值。

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> 当您使用 QLDB 控制台或 QLDB Shell 查询表 ID 或文档 ID（或任何字符串文字值）时，返回的值用双引号括起来。但是，在指定REDACT_REVISION存储进程的表 ID 和文档 ID 参数时，必须用单引号将这些值括起来。
>
> 这是因为您以 PartiQL 格式编写语句，但 QLDB 会以 Amazon Ion 格式返回结果。有关 QLDB 中 PartiQL 的语法和语义详细信息，请参阅使用 PartiQL 查询 Ion。

有效的编校请求会返回一个 Ion 结构，该结构表示您正在编校的文档修订版，如下所示。

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

通过运行此存储过程后，QLDB 将异步处理的编校请求。编校完成后，该data 结构将被永久删除，并由一个新 *dataHash* 字段取而代之。该字段的值是已移除 data 结构的 Ion 哈希，如下所示。

> **Note**
>
> 此 dataHash 示例仅供参考，不是实际计算的哈希值。

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

删除和编校当前修订版本

有效文档修订版（即每个文档中未删除的最新修订版）不符合数据编校的条件。必须先更新或删除当前修订版本，然后才能对其进行编校。这会将先前处于活动状态的修订版本移至历史记录，使其有资格进行编校。

如果您的用例要求将整个文档标记为已删除，则首先使用DELETE语句。例如，以下语句在逻辑上删除VIN为1HVBBAANXWH544237的VehicleRegistration文档。

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

然后，如前所述，在删除之前编校之前的修订版。如果需要，您也可以单独编校任何先前的修订版。

如果您的用例要求文档保持活动状态，则首先使用 UPDATE 或 FROM 语句来掩盖或删除要编校的字段。此过程将在以下部分中介绍。

编校修订版本中的特定字段

QLDB 不支持在文档修订版中编校特定字段。为此，您可以先使用UPDATE-REMOVE或 FROM-REMOVE语句从修订版中移除现有字段。例如，使用以下语句移除 VIN 为1HVBBAANXWH544237的VehicleRegistration文档中的LicensePlateNumber字段。

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

然后，如前所述，在删除之前编校之前的修订版。如果需要，您还可以单独编校包含此现已删除字段的任何先前修订版。

要了解如何优化查询，请继续 优化查询性能。

优化查询性能

Amazon QLDB 旨在满足高性能联机事务处理 (OLTP) 的工作负载需求。这意味着 QLDB 已针对一组特定查询模式进行了优化，尽管它支持类似 SQL 的查询功能。涉及应用程序及其数据模型对于使用此查询模式至关重要。否则，随着表的增长，您将遇到严重的性能问题，包括查询延迟、事务超时和并发冲突。

该章节描述了 Amazon QLDB 中的查询限制，并提供了在这些限制条件下优化查询性能指导。

主题

- 事务超时限制
- 并发冲突
- 最优查询模式
- 要避免的查询模式
- 监控性能

事务超时限制

QLDB 中的每个 PartiQL 语句（包括每个 SELECT 查询）都是在事务中处理的，并且受事务超时限制的约束。在提交之前，事务最多可运行 30 秒。超过此限制后，QLDB 会拒绝在事务完成的任何工作，并丢弃运行该事务会话。此限制通过启动事务（而不提交或取消事务）以保护服务的客户端免遭泄漏会话。

并发冲突

在 QLDB 中，并发控制通过乐观并发控制 (OCC) 实现。次优查询也可能导致更多 OCC 冲突。有关 OCC 的信息，请参阅Amazon QLDB 并发模型。

最优查询模式

按最佳实践标准，您应运行带有WHERE谓词子句的语句，该子句可以筛选索引字段或文档 ID。QLDB 需要在索引字段上使用相等运算符（例如=或IN)才能高效地查找文档。

以下是 用户视图 中最佳查询模式的示例。

```
--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'
```

任何不遵循这些模式的查询都会调用 全表扫描。表扫描可能会导致大型表上的查询，或返回大型结果集的查询发生事务超时。它们还可能导致 OCC 与竞争事务发生冲突。

高基数索引

我们建议为包含高基数值的字段建立索引。例如，VehicleRegistration表中的 VIN 和 LicensePlateNumber 字段是索引字段，旨在保持唯一性。

避免对低基数字段（例如状态代码、地址州或省以及邮政编码）建立索引。如果您为此类字段编制索引，则查询可能会生成较大的结果集，这些结果集 更有可能导致事务超时或导致意外的 OCC 冲突。

已提交视图查询

您在已提交视图中运行的查询遵循与用户视图查询相同的优化准则。您在表上创建的索引也可用于已提交视图中的查询。

历史记录函数查询

历史记录函数查询不使用您在表上创建的索引。QLDB 历史记录按文档 ID 编制索引，目前无法创建其他历史索引。

最佳做法是，使用日期范围（开始时间 和 结束时间）和文档 ID (`metadata.id`) 来限定历史查询。包含开始时间和结束时间的历史记录查询将从日期范围限定中获得便利。

内部联接查询

对于内部联接查询，请使用至少包含联接右侧表的索引字段的联接条件。如果没有连接索引，连接查询将调用多个表扫描 - 对于连接左表中的每个文档，查询将完全扫描右表。除了为至少一个表指定`WHERE`相等谓词外，最佳做法是对要加入的每个表建立索引的字段进行联接。

例如，以下查询将 `VehicleRegistration`和`Vehicle`表连接至各自的字段，这两个`VIN`字段均已编制索引。此查询还有一个`VehicleRegistration.VIN`相等谓词。

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

为联接查询中的联接条件与相等谓词选择高基数索引。

要避免的查询模式

以下是次优语句的示例，这些语句不能很好地扩展 QLDB 中较大的表。我们强烈建议您不要依赖这些类型的查询来获取随时间增长的表，因为您的查询最终将导致事务超时。由于表包含大小不同的文档，因此很难为非索引查询定义精确的限制。

```
--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)
```

通常，我们不建议在 QLDB 中为生产用例运行以下类型查询模式：

- 联机分析处理 (OLAP) 查询
- 没有谓词子句的探索性查询
- 报告查询
- 文本搜索

相反，我们建议将您的数据流式传输到专门构建的数据库服务，该服务针对分析用例进行了优化。例如，您可以将 QLDB 数据流式传输至 Amazon OpenSearch Service，以提供文档全文搜索功能。有关演示此用例的示例应用程序，请参阅 GitHub 存储库 aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python。有关 QLDB 流的信息，请参阅 Amazon QLDB 流式传输日记账数据。

监控性能

QLDB 驱动程序在语句结果对象中提供消耗的 I/O 使用情况和计时信息。您可使用这些指标识别效率低下的 PartiQL 语句。要了解更多信息，请继续 获取 PartiQL 语句统计信息。

您还可以使用 Amazon CloudWatch 跟踪分类账的数据操作性能。监控指定`LedgerName` 和 `CommandType`的`CommandLatency`指标。有关更多信息，请参阅使用 Amazon 进行监控 CloudWatch。要了解 QLDB 如何使用命令管理数据操作，请参阅驱动程序会话管理。

获取 PartiQL 语句统计信息

Amazon QLDB 提供语句执行统计信息，可帮助您通过运行更高效的 PartiQL 语句来优化 QLDB 的使用。QLDB 将这些统计信息与语句结果一起返回。它们包含量化消耗的 I/O 使用情况和服务器端处理时间的指标，您可以使用这些指标来识别低效的语句。

此功能当前用于QLDB 控制台、 QLDB Shell以及最新版 QLDB 驱动程序（所有语言）的PartiQL 编辑器。您还可在控制台上查看查询历史的语句统计信息。

主题

I/O 使用率

I/O 使用率指标介绍了读取 I/O 请求的数量。如果读取 I/O 请求的数量高于预期，则表示语句未经优化，例如缺少索引。我们建议您在上一主题 优化查询性能 中复习最优查询模式。

> **Note**
>
> 在非空表上运行`CREATE INDEX`语句时，I/O 使用率指标仅包含同步索引创建调用的读取请求。
>
> QLDB 异步为表中的任何现有文档创建索引。这些异步读取请求不包括在语句结果的 I/O 使用率指标中。异步读取请求单独收费，且在索引构建完成后计入您的读取 I/O 总数。

使用 QLDB 控制台

若要使用 QLDB 控制台获取语句的读取 I/O 使用情况，请执行以下步骤：

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 PartiQL 编辑器。
3. 从分类账下拉列表中选择分类帐。
4. 在查询编辑器窗口中，输入您选择的任何语句，然后选择 运行。以下是查询示例。

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   要运行语句，你可对 Windows 使用快捷键Ctrl+Enter，对 macOS 使用Cmd+Return。有关更多键盘快捷键的信息，请参阅 PartiQL 编辑器键盘快捷键。
5. 在查询编辑器窗口下方，您的查询结果包括读取 I/O，此语句发出的读取请求数。

您还可通过执行以下步骤来查看查询历史记录的读取 I/O：

1. 在导航窗格中，选择PartiQL 编辑器下的最新查询。
2. 读取 I/O 列显示每条语句发出的读取请求数。

使用 QLDB 驱动程序

要使用 QLDB 驱动程序获取语句 I/O 使用情况，请调用结果的流指针或缓冲指针的`getConsumedIOs`操作。

以下代码示例显示如何从语句结果的流指针 中读取 I/O。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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))
```

以下代码示例显示如何从语句结果的缓冲指针中读取 I/O。这将返回来自 `ExecuteStatement` 和`FetchPage` 请求的读取 I/O 总数。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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**
>
> 流指针是有状态的，因为它会对结果集分页。因此，`getConsumedIOs` 和`getTimingInformation`操作会返回您调用这些指标时的累积指标。缓冲后的指针将结果集缓存至内存中，并返回累积的指标总数。

计时信息

计时信息指标描述了服务器端处理时间（以毫秒为单位）。服务器端处理时间定义为 QLDB 在处理语句上所需时间。这不包括网络通话或暂停时间。该指标将 QLDB 服务端的处理时间与客户端处理时间区分开来。

使用 QLDB 控制台

要使用 QLDB 控制台获取语句时序信息，请执行以下步骤：

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 PartiQL 编辑器。
3. 从分类账下拉列表中选择分类帐。
4. 在查询编辑器窗口中，输入您选择的任何语句，然后选择 运行。以下是查询示例。

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   要运行语句，你可对 Windows 使用快捷键Ctrl+Enter，对 macOS 使用Cmd+Return。有关更多键盘快捷键的信息，请参阅 PartiQL 编辑器键盘快捷键。

5. 在查询编辑器窗口下方，您的查询结果包括服务器端延迟，即 QLDB 收到语句请求与发送响应之间的时间长度。这是总查询时长子集。

您还可通过执行以下步骤来查看查询历史记录的计时信息：

1. 在导航窗格中，选择PartiQL 编辑器下的最新查询。
2. 执行时间 (ms)列将会显示每条语句的计时信息。

使用 QLDB 驱动程序

要使用 QLDB 驱动程序获取语句的计时信息，请调用结果的流指针或缓冲指针的`getTimingInformation`操作。

以下代码示例显示如何从语句结果的流指针 中获取处理时间。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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))
```

以下代码示例显示如何从语句结果的 缓冲指针 中获取处理时间。这将返回来自 ExecuteStatement和FetchPage 请求的总处理时间。

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**
>
> 要转换为同步代码，请移除await和async关键字，然后将 IAsyncResult类型更改为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**
>
> 流指针是有状态的，因为它会对结果集分页。因此，`getConsumedIOs` 和`getTimingInformation`操作会返回您调用这些指标时的累积指标。缓冲后的指针将结果集缓存至内存中，并返回累积的指标总数。

要了解如何查询系统目录，请继续 查询系统目录。

查询系统目录

您在 Amazon QLDB 分类账中创建的每个表都有一个系统分配的唯一 ID。您可通过查询系统目录表`information_schema.user_tables`来查找表的 ID、其索引列表和其他元数据。

所有系统分配的 ID 都是通用唯一标识符 (UUID)，每个标识符都以 Base62 编码的字符串表示。有关更多信息，请参阅Amazon QLDB 中的唯一编号。

以下示例显示返回 VehicleRegistration 表元数据属性的查询结果。

```
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"
}
```

表的元数据字段

- tableId – 表的唯一 ID。
- name – 表的名称。
- indexes — 表中的索引列表。
  - indexId – 索引的唯一 ID。
  - expr — 已编入索引的文档路径。该字段采用的字符串格式为：[fieldName]。
  - status — 索引的当前状态（BUILDING、FINALIZING、ONLINE、FAILED、或DELETING）。在状态为ONLINE之前，QLDB 不会在查询中使用该索引。
  - message — 描述索引FAILED处于状态的原因的错误消息。仅在失败的索引中包含此字段。
- status — 表格的当前状态（ACTIVE或INACTIVE）。当您DROP时表格成为INACTIVE。

要了解如何使用 DROP TABLE 和 UNDROP TABLE 语句管理表，请继续管理表。

管理表

本节介绍如何在 Amazon QLDB 中使用 DROP TABLE 和 UNDROP TABLE 语句管理表。它还描述了如何在创建表时对其标记。您可以创建的活动表数量和表总数的限额在Amazon QLDB 资源中的限额和限制中定义。

主题

创建时对表格进行标记

> **Note**
>
> 只有 STANDARD 权限模式分类账支持在创建时对表格进行标记。

您可以标记您的资源。要管理现有表标签，请使用AWS Management Console或 API 操作TagResource、UntagResource、和ListTagsForResource。有关更多信息，请参阅为 Amazon QLDB 资源贴标签。

您还可以在创建表时使用 QLDB 控制台或在 CREATE TABLEPartiQL 语句中指定表标签来定义表标签。下面的示例创建了一个名为 Vehicle 的表，带有标记 environment=production。

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

通过在创建资源时对其进行标记，无需在创建资源后运行自定义标记脚本。标记表后，您可根据这些标签来控制对表的访问。例如：您只能向具有特定标签的表授予完全访问权限。有关 JSON 策略示例，请参阅基于表格标签对所有操作的完全访问权限。

删除表格

要删除表，请使用基本 DROP TABLE 语句。当你在 QLDB 中删除表格时，只是将其停用。

例如，以下语句使VehicleRegistration表处于停用状态。

```
DROP TABLE VehicleRegistration
```

DROP TABLE语句返回系统分配的表的 ID。状态VehicleRegistration应该INACTIVE在系统目录表 information_schema.user _tables 中。

```
SELECT status FROM information_schema.user_tables
```

```
WHERE name = 'VehicleRegistration'
```

查询非活动表历史记录

除了表名之外，您还可以使用表 ID 作为第一个输入参数查询QLDB历史记录函数 。必须使用表 ID 查询非活动表的历史记录。停用表后，您将无法再使用表名查询历史记录。

首先，通过查询系统目录表查找表 ID。例如，以下查询将返回 VehicleRegistration 表的 tableId。

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

然后，您可使用此 ID 从查询修订历史记录中运行相同的历史记录查询。以下是从表 ID5PLf9SXwndd63lPaSIa0O6 中查询文档 IDADR2Ll1fGsU4Jr4EqTdnQF 历史记录的示例。表 ID 为字符串文本值，必须括至单引号内。

```
--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'
```

重新激活表格

在 QLDB 中停用表后，可以使用取消删除表语句将其重新激活。

首先，从information_schema.user_tables中找到表 ID。例如，以下查询将返回 VehicleRegistration 表的 tableId。状态应为 INACTIVE。

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

然后使用此 ID 重新激活表。以下是取消表 ID5PLf9SXwndd63lPaSIa0O6 的示例。此时，表 ID 是用双引号括起的唯一标识符。

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

现在的VehicleRegistration状态应该是ACTIVE。

若要了解如何创建、描述和删除索引，请继续 管理索引。

管理索引

本部分介绍如何在 Amazon QLDB 中创建、描述和删除索引。可以为每个表创建的索引数量的限额在Amazon QLDB 资源中的限额和限制中定义。

主题

创建索引

如创建表与索引中所述，您可以使用 CREATE INDEX 语句在表上为指定的顶级字段创建索引，如下所示。表名和索引字段名称均为区分大小写。

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

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

您在表上创建的每个索引都包含一个系统分配的唯一 ID。若要查找此索引 ID，请参阅以下部分 描述索引。

> **Important**
>
> QLDB 需要索引才能高效查找文档。如果没有索引，QLDB 在读取文档时需进行全表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (=或IN) 运行带有WHERE谓词子句的语句。有关更多信息，请参阅优化查询性能。

创建索引时应注意以下限制：

- 只能在单个顶级字段创建索引。不支持复合索引、嵌套索引、唯一索引以及基于函数的索引。
- 您可以为任何 Ion 数据类型创建索引，其中包括 `list` 和 `struct`。但是，无论 Ion 类型如何，您都只能通过整个 Ion 值进行索引查找。例如，使用 `list` 类型作为索引时，不能按列表中的一个项目进行索引查找。

- 只有使用相等谓词时，查询性能才会得到改善；例如`WHERE indexedField = 123` 或 `WHERE indexedField IN (456, 789)`。

  QLDB 不支持查询谓词不等式。因此，未实施范围过滤扫描。
- 索引字段名称区分大小写，且最大长度可为 128 个字符。
- 在 QLDB 中创建索引具有异步特点。非空表上完成索引所需的时间取决于表的大小。有关更多信息，请参阅管理索引。

描述索引

在 QLDB 中创建索引具有异步特点。非空表上完成索引所需的时间取决于表的大小。要检查索引构建的状态，可以查询系统目录表information_schema.user_tables。

例如，以下语句在系统目录中查询 `VehicleRegistration` 表上的所有索引。

```
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"
}
```

索引字段数

- `indexId` – 索引的唯一 ID。
- `expr` — 已编入索引的文档路径。该字段采用的字符串格式为：`[fieldName]`。
- `status` — 索引的当前状态。索引的状态可以是以下值之一：
  - `BUILDING` — 正在积极为表建立索引。

- FINALIZING — 已完成索引的构建并开始激活它以供使用。
- ONLINE — 处于活动状态，可在查询中使用。在状态为在线之前，QLDB 不会在查询中使用该索引。
- FAILED— 由于出现不可恢复的错误，无法建立索引。此状态的索引仍计入每个表的索引限额。有关更多信息，请参阅常见错误。
- DELETING — 用户删除索引后正在主动删除索引。

- message — 描述索引FAILED处于状态的原因的错误消息。仅在失败的索引中包含此字段。

使用控制台

您还可以使用AWS Management Console检查索引的状态。

检查索引的状态（控制台）

1. 登录 AWS Management Console 并打开 Amazon QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择分类账。
3. 在分类帐列表中，选择要管理其索引的分类帐名称。
4. 在分类账详细信息页面的表格选项卡下，选择要检查其索引的表名。
5. 在表格详细信息页面上，找到 已编入索引的字段 卡片。索引状态 列显示表中每个索引的当前状态。

删除索引

使用 DROP INDEX 语句删除索引。删除索引时，该索引会从表中永久删除。

首先，从information_schema.user_tables中找到索引 ID。例如，以下查询返回VehicleRegistration表中已编入索引LicensePlateNumber字段的indexId。

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

然后，使用此 ID 删除索引。以下是取消表 ID 4tPW3fUhaVhDinRgKRLhGU的示例。此时，索引 ID 是用双引号括起的唯一标识符。

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> 该子句 `WITH (purge = true)` 是所有 `DROP INDEX` 语句所必需的，并且 `true` 是目前唯一支持的值。
> 关键字purge区分大小写，并且必须为全小写。

使用控制台

您也可以使用 AWS Management Console 删除索引。

删除索引（控制台）

1. 登录 AWS Management Console 并打开 Amazon QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择分类账。
3. 在分类帐列表中，选择要管理其索引的分类帐名称。
4. 在分类账详细信息页面的表格选项卡下，选择要删除其索引的表名。
5. 在表格详细信息页面上，找到 已编入索引的字段 卡片。选择要删除的索引，然后选择 删除索引。

常见错误

本节介绍创建索引时可能遇到的常见错误，并建议可能的解决方案。

> **Note**
>
> 处于这种`FAILED`状态的索引仍计入每个表的索引限额。失败的索引还会阻止您修改或删除导致在表上创建索引失败的任何文档。
> 您必须明确删除 索引才能将其从限额中删除。

文档包含索引字段的多个值：字段 **##**。

QLDB 无法为指定字段名建立索引，因为该表包含一个文档，该文档具有同一个字段的多个值（即重复的字段名）。

必须先删除失败索引。然后，在重试创建索引前，请确保表中的所有文档的每个字段名只有一个值。您也可以为没有重复项的字段创建索引。

如果您尝试插入的文档包含已在表格上编制索引字段的多个值，QLDB 也会返回此错误。

已超过索引限制：***TableName* ##** n 个索引，因此无法创建更多索引。

QLDB 强制每个表最多只能包含五个索引，包括失败的索引。在创建新索引之前，必须要删除现有索引。

为定义以下标识符索引：***indexId***。

您试图删除指定的表和索引 ID 组合中不包含的索引。若要了解如何检查现有索引，请参阅描述索引。

Amazon QLDB 中的唯一编号

本节介绍了 Amazon QLDB 中系统分配唯一标识符的属性和使用指南。它还提供了 QLDB 唯一 ID 的部分示例。

主题

- 属性
- 用量
- 示例

属性

QLDB 中所有系统分配的 ID 均为通用唯一标识符 (UUID)。每个 ID 具有以下属性：

- 128 位 UUID 号
- 以 Base62 编码的文本表示
- 22 个字符的固定长度字母数字的字符串（例如`3Qv67yjXEwB9SjmvkuG6Cp`）

用量

当应用程序中使用 QLDB 唯一 ID 时，请注意以下指南：

Do

- 将 ID 视为字符串。

切勿

- 尝试解码字符串。
- 将语义含义归因于字符串（如推导时间分量）。
- 按语义顺序对字符串排序。

示例

以下属性是 QLDB 中唯一 ID 的部分示例：

- 文档 ID
- 索引 ID
- Strand ID
- 表 ID
- 事务 ID

Amazon QLDB 并发模型

Amazon QLDB 旨在满足高性能联机事务处理 (OLTP) 的工作负载需求。QLDB 支持类似 SQL 的查询功能，并提供完整 ACID 事务。此外，QLDB 数据项为文档，可提供架构灵活性和直观的数据建模。以日志为核心，您可以使用 QLDB 访问数据所有更改的完整且可验证的历史记录，并根据需要将连贯的事务流式传输到其他数据服务。

主题

乐观并发控制

在 QLDB 中，并发控制通过乐观并发控制 (OCC) 实现。OCC 的运作原则是，多笔事务可以在不相互干扰的情况下经常完成。

通过 OCC，QLDB 中的事务不会获取数据库资源的锁，并且在完全可序列化的隔离下运行。QLDB 以串行方式运行并发事务，因此它产生的效果与串行启动事务的效果相同。

在提交之前，每个事务都会执行验证检查，以确保没有其他已提交的事务修改其正在访问的数据。如果此检查发现冲突的修改，或者数据的状态发生更改，则提交事务将被拒绝。但是，事务处理可以重新启动。

当事务写入 QLDB 时，OCC 模型的验证检查由 QLDB 本身实现。如果由于OCC的验证阶段失败而无法将事务写入日志，QLDB 将 `OccConflictException`返回到应用层。应用软件负责确保事务重新启动。应用程序应中止被拒绝的事务，然后从头开始重试整个事务。

要了解 QLDB 驱动程序如何处理和重试 OCC 冲突和其他临时异常，请参阅使用 Amazon QLDB 中的驱动程序了解重试策略。

使用索引避免全表扫描

QLDB 中的每个 PartiQL 语句（包括每个 SELECT 查询）都是在事务中处理的，并且受事务超时限制的约束。

按最佳实践标准，您应运行带有WHERE谓词子句的语句，该子句可以筛选索引字段或文档 ID。QLDB 需要在索引字段上使用相等运算符才能高效地查找文档；例如WHERE indexedField = 123或WHERE indexedField IN (456, 789)。

如果没有索引查询，QLDB 在读取文档时需进行 全表扫描。这可能会导致查询延迟和事务超时，还会导致 OCC 与竞争事务发生冲突的机会增加。

例如，假设一个名为Vehicle的表，该表仅在该VIN字段上有索引。它将包含以下文档。

VIN	Make	模型	颜色
"1N4AL11D75C109151"	"Audi"	"A5"	"Silver"
"KM8SRDHF6EU074761"	"Tesla"	"Model S"	"Blue"
"3HGGK5G53FM761765"	"Ducati"	"Monster 1200"	"Yellow"
"1HVBBAANXWH544237"	"Ford"	"F 150"	"Black"
"1C4RJFAG0FC625797"	"Mercedes"	"CLK 350"	"White"

两个名为 Alice 和 Bob 的并发用户正在使用分类账中的同一个表。以下两人想要更新两个不同的文档，如下所示。

Alice：

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

Bob：

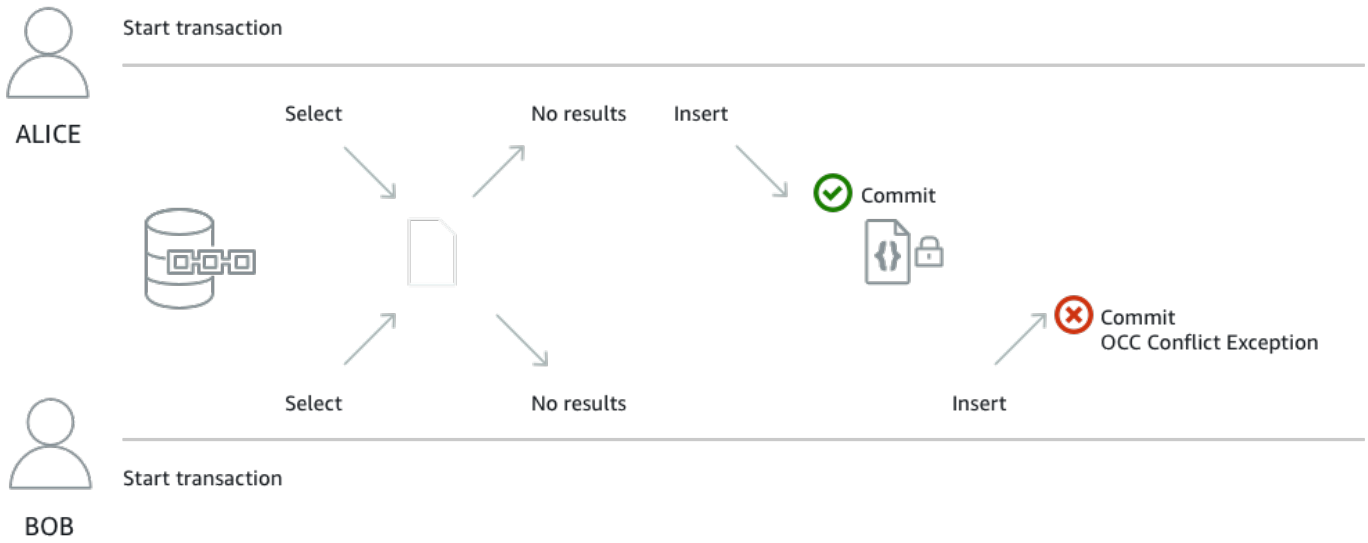
```
UPDATE Vehicle AS v
SET v.Color = 'Red'
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

假设 Alice 和 Bob 同时开始事务。Alice 的UPDATE语句对VIN字段进行索引查找，因此它只需要读取那个文档即可。Alice 先完成并成功提交了她的事务。

Bob 的语句会筛选非索引字段，因此它会进行表扫描并遇到OccConflictException。这是因为 Alice 提交的事务修改了 Bob 语句正在访问的数据，其中包括表中的每个文档，而非仅仅是 Bob 正在更新的文档。

插入 OCC 冲突

OCC 冲突可能包括新插入文档，而不仅仅是先前存在的文档。请考虑下图，其中两位用户(Alice 和 Bob)正在处理 表中的同一项目。他们都希望仅在谓词值尚不存在的情况下插入新文档。



在此示例中，Alice 和 Bob 在单个事务中运行以下 SELECT 和 INSERT语句。只有当INSERT 语句未返回任何结果时，他们的应用程序才会运行该SELECT 语句。

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
```

```
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

假设 Alice 和 Bob 同时开始事务。他们的两个 SELECT 查询都没有返回任何带有ABCDE12345EXAMPLE为VIN的现有文档。因此，他们的应用程序继续执行 INSERT 语句。

Alice 先完成并成功提交了她的事务。然后，Bob 尝试提交他的事务，但是 QLDB 拒绝了事务并抛出了OccConflictException。这是因为 Alice 提交的事务修改了 Bob SELECT查询结果集，而 OCC 在提交 Bob 的事务之前就检测到了这种冲突。

此事务示例需要SELECT查询才能具有幂等。然后 Bob 可从一开始就重试整个事务。但是他的下一个SELECT查询将返回 Alice 插入的文档，因此 Bob 的应用程序将无法运行 INSERT。

使事务幂等

上一节中的插入事务也是幂等事务的示例。换句话说，多次运行同一个事务会产生相同结果。如果 Bob 在不事先检查特定INSERT是否VIN存在的情况下运行，则该表最后可能会出现具有重复VIN值的文档。

除 OCC 冲突之外，请考虑其他重试方案。例如，假设 QLDB 在服务器端成功提交了事务，但客户端在等待响应时超时。作为最佳做法，我们建议将写事务设置为幂等，以避免在并发或重试时出现任何意想不到的副作用。

Redcation OCC 冲突

QLDB 可防止在同一日志块同时编辑修订版。举一个例子，其中两个并发用户 (Alice 和 Bob) 想要编辑在分类账的同一个区块上提交的两个不同的文档修订版。首先，Alice 通过运行 REDACT_REVISION 存储过程请求编辑一个修订版，如下所示。

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

然后，当 Alice 的请求仍在处理，Bob 请求编辑另一个修订版，如下所示。

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB 拒绝了 Bob 的请求，`OccConflictException`尽管他们正试图编辑两个不同的文档修订版。这是因为 Bob 的修订版与 Alice 正编辑的修订版位于同一个区块上。Alice 的请求处理完毕后，Bob 可以重试他的编辑请求。

同样，如果两个并发事务尝试编辑同一修订版，则只能处理一个请求。在编辑完成之前，另一个请求失败并显示 OCC 冲突异常。之后，任何对同一修订版本进行编辑的请求都会导致错误，表明该修订已被编辑。

管理并发会话

如您有使用关系数据库管理系统 (RDBMS) 的经验，可能已熟悉并发连接限制。QLDB 与传统 RDBMS 连接的概念不同，因为事务通过 HTTP 请求和响应消息运行。

在 QLDB 中，类比概念是活跃会话。从概念上讲，会话类似于用户登录，它管理有关您对分类账的数据事务请求的信息。活动会话是指正在积极运行事务会话。它可以是最近完成事务的会话，服务预计它将立即启动另一笔事务。QLDB 支持每个会话正在运行的事务。

每个分类账的并发活动会话限制在Amazon QLDB 资源中的限额和限制中定义。达到此限制后，任何尝试启动事务的会话都会导致错误 (`LimitExceededException`)。

有关会话生命周期以及 QLDB 驱动程序在运行数据事务时如何处理会话的信息，请参阅驱动程序会话管理。有关使用 QLDB 驱动程序在应用程序中配置会话池的最佳实践，请参阅Amazon QLDB 驱动程序建议中的配置 QldbDriver 对象。

Amazon QLDB 中的数据验证

借助 Amazon QLDB，您可以相信对应用程序数据的更改历史是准确无误的。QLDB 使用不可变的事务日记账（称为日记账）进行数据存储。日记账会跟踪已提交数据的每一次更改，并保存完整、可验证的更改历史记录。

QLDB 使用 SHA-256 哈希函数和基于 Merkle 树的模型来生成日记账的加密表示，即摘要。摘要是数据在某个时间点上整个更改历史的唯一签名。您可以使用摘要来验证您的文档修订相对于该签名的完整性。

主题

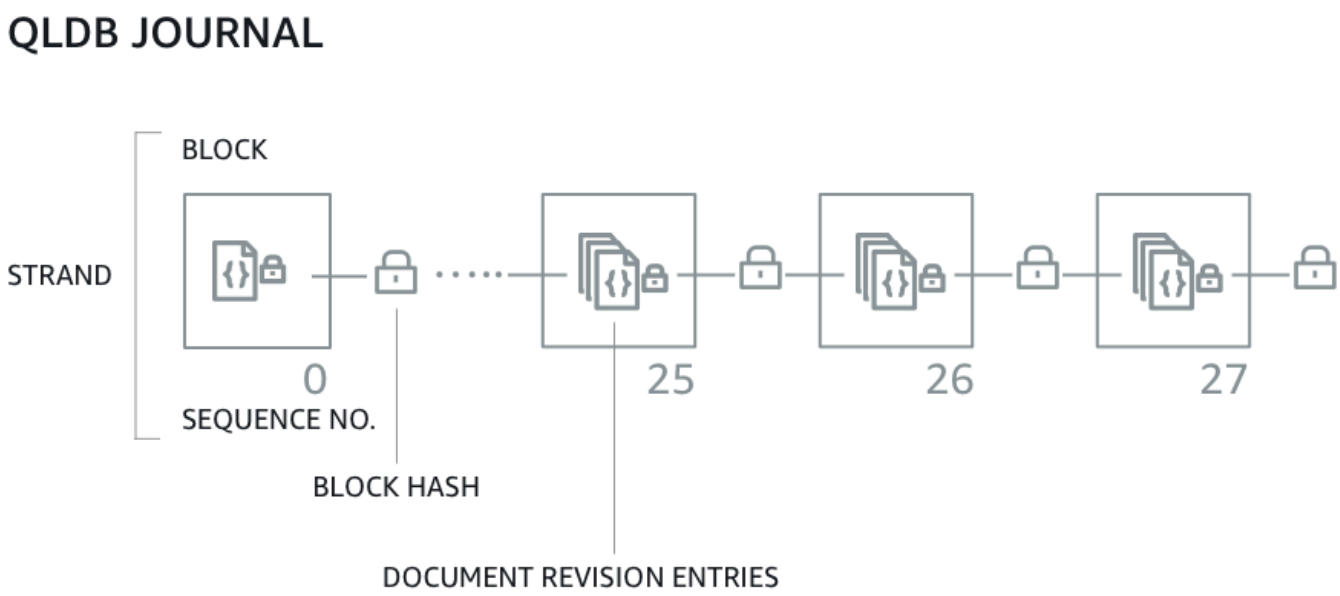
您可以在 QLDB 中验证哪种数据？

在 QLDB 中，每个分类账只有一个日记账。一个日记账可以有多个链，即日记账的分区。

> **Note**
>
> QLDB 目前仅支持单链日记账。

区块是在事务过程中提交到日记账链的对象。此区块包含条目对象，代表事务产生的文档修订。您可以在 QLDB 中验证单个修订或整个日记账区块。

下面的示意图阐明了此日记账结构。



该图显示事务作为包含文档修订条目的数据块提交到日记账中。它还显示，每个区块都与后续区块进行了散列，并有一个序列号来指定其在链中的地址。

有关块中数据内容的更多信息，请参阅 Amazon QLDB 中的日记账内容。

数据完整性意味着什么？

QLDB 中的数据完整性意味着您的分类账日记账实际上是不可变的。换句话说，您的数据（特别是每个文档修订）处于以下情况的状态：

1. 它存在于日记账中最初写下它的位置。
2. 自撰写以来，它从未被以任何方式修改过。

验证是如何运行的？

要了解验证在 Amazon QLDB 中的工作原理，您可以将该概念分解为四个基本组成部分。

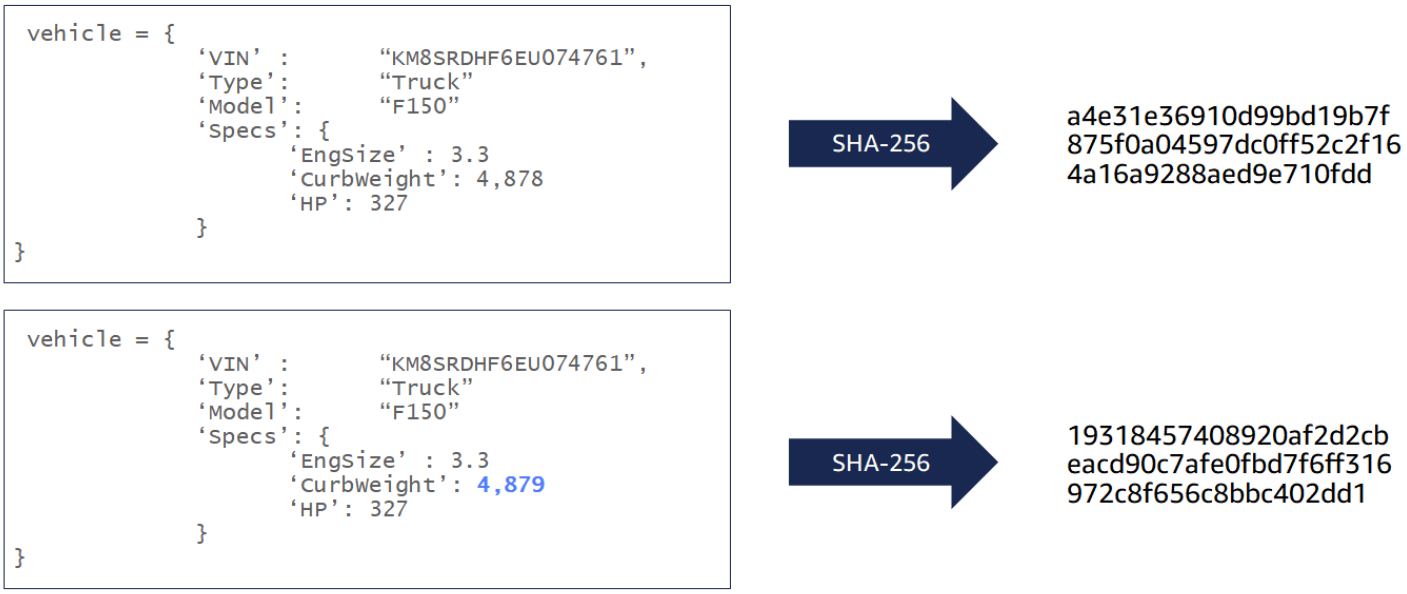
- 哈希
- 摘要
- Merkle 树

- 证明

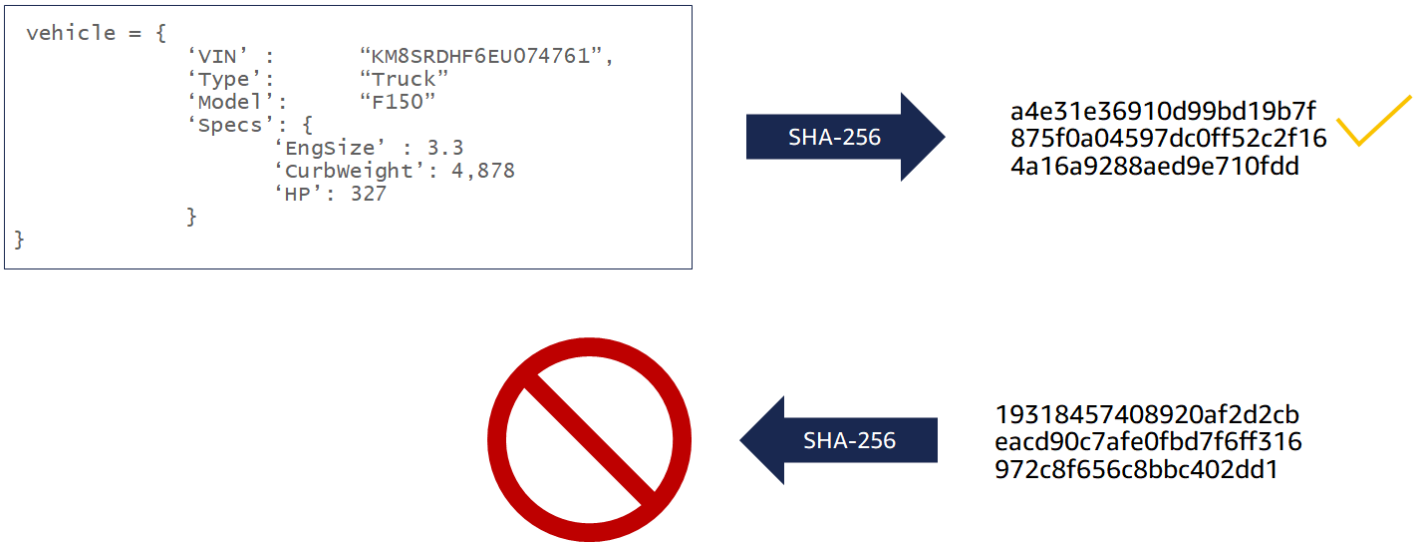
哈希

QLDB 使用 SHA-256 加密哈希函数创建 256 位的哈希值。哈希值是任意数量输入数据的唯一固定长度签名。如果更改输入的任何部分，哪怕是一个字符或一个比特，那么输出哈希值就会完全改变。

下图显示，SHA-256 哈希函数为两个 QLDB 文档创建完全唯一的哈希值，而这些哈希值仅相差一个数字。



SHA-256 散列函数是单向的，这意味着在给定输出的情况下计算输入在数学上是不可行的。下图显示，在给定输出哈希值时，计算输入 QLDB 文档是不可行的。



出于验证目的，以下数据输入在 QLDB 中经过哈希处理：

- 文档修订
- PartiQL 语句
- 修订条目
- 日记账区块

摘要

摘要是您分类帐在某一时刻的整个日记账的加密表示。日记账是只追加的，日记账区块按序号排列并像区块链一样进行哈希链接。

您可以随时请求分类账摘要。QLDB 生成摘要并将其作为安全输出文件返回给您。然后，您使用该摘要来验证在先前时刻提交的文档修订的完整性。如果您通过从一个修订开始重新计算哈希直到摘要，您可以证明在此期间数据未被更改。

Merkle 树

随着您的分类帐规模的增长，重新计算整个日记账的哈希链进行验证变得越来越低效。QLDB 使用 Merkle 树模型来解决这种效率低下的问题。

Merkle 树是一种树数据结构，其中每个叶子节点代表一个数据块的哈希值。每个非叶子节点都是其子节点的哈希值。Merkle 树通常用于区块链，它通过一个审计证明机制，帮助您高效地验证大型数据集。有关 Merkle 树的更多信息，请参阅 Merkle 树维基百科页面。要了解有关 Merkle 审计证明的更多信息以及示例用例，请参阅证书透明度网站上的日记账证明的工作原理部分。

Merkle 树的 QLDB 实现是由日记的完整哈希链构造的。在此模型中，叶节点是所有单个文档修订哈希的集合。根节点表示截至某个时间点的整个日记账的摘要。

使用 Merkle 审计证明，您可以通过仅检查分类账修订历史记录的一小部分来验证修订。通过从给定的叶节点（修订）遍历树到其根节点（摘要），您可以实现这一点。沿着这条遍历路径，您递归地对节点的兄弟对进行哈希运算，直到最终得到摘要。这种遍历具有树中`log(n)`节点的时间复杂度。

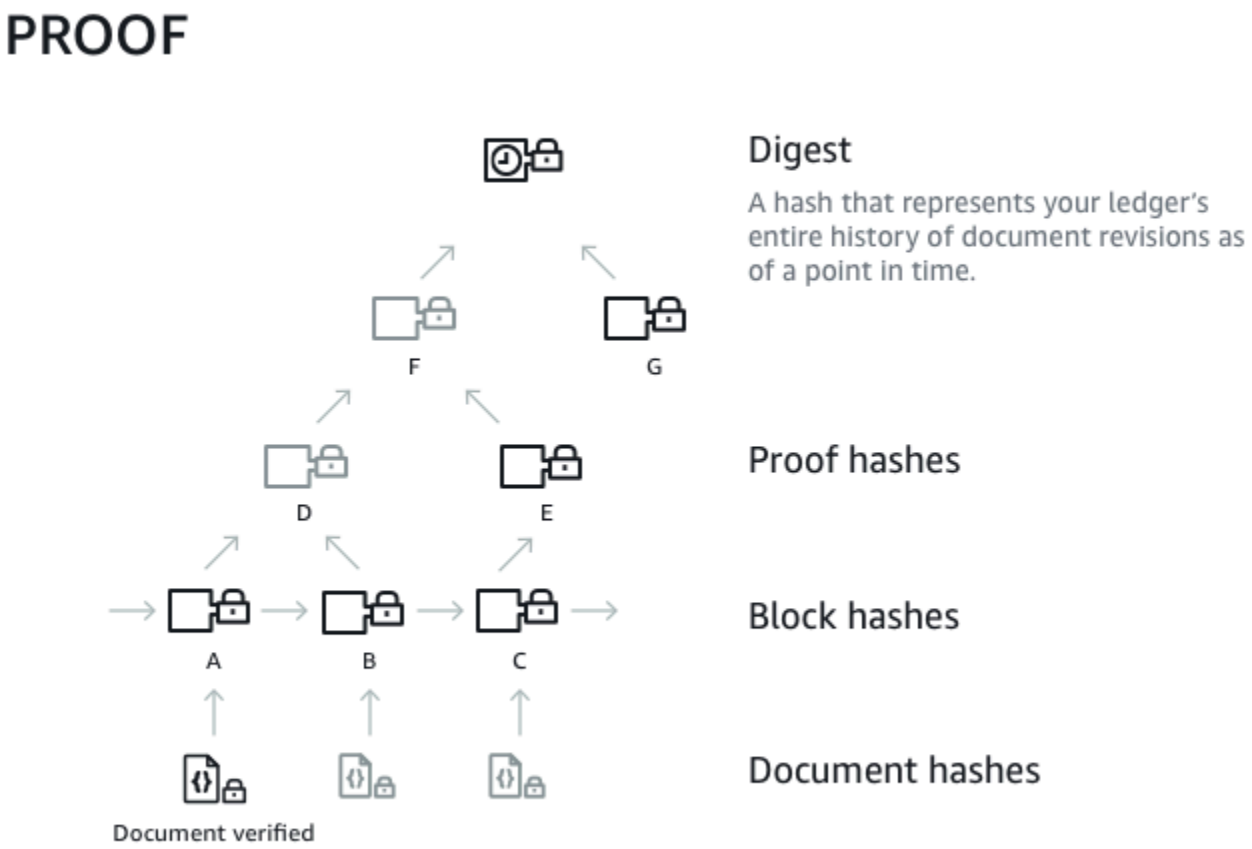
证明

证明是 QLDB 针对给定摘要和文档修订返回的节点哈希的有序列表。它由 Merkle 树模型所需的哈希，用于将给定的叶节点哈希（一个修订）链接到根哈希（摘要）。

在修订和摘要之间更改任何已提交的数据会破坏日记账的哈希链，并导致无法生成证明。

验证示例

下图阐明了 Amazon QLDB 哈希树模型。它显示了一组向上滚动到顶部根节点的区块哈希，它代表了日记账链的摘要。在具有单链日记账的分类账中，这个根节点也是整个分类账的摘要。



假设节点 A 是包含您要验证哈希的文档修订的块。以下节点代表 QLDB 在您的证明中提供的有序哈希列表：B、E、G。 这些哈希值是为了从哈希 A 重新计算摘要而必需的。

要重新计算摘要，请执行以下操作：

1. 从哈希 A 开始，然后将其与哈希 B 连接起来。 然后，对结果进行哈希处理以计算 D。
2. 使用 D 和 E 来计算 F。
3. 使用 F 和 G 来计算摘要。

如果您重新计算的摘要与预期值匹配，那么验证就是成功的。鉴于一个修订哈希和一个摘要，不可行对证明中的哈希进行逆向工程。因此，这个过程证明了相对于摘要，确实将您的修订写入了这个日记账位置。

数据编辑对验证有何影响？

在 Amazon QLDB，DELETE 语句只能通过创建将文档标记为已删除的新修订版，从逻辑上删除文档。QLDB 还支持数据编校操作，允许您永久删除表历史记录中的非活动文档修订版本。

密文操作仅删除指定修订中的用户数据，而日记账序列和文档元数据则保持不变。修订已被编辑后，修订中的用户数据（由 data 结构表示）将替换为一个新 dataHash 字段。该字段的值是已移除 data 结构的 Amazon Ion 哈希。有关修订操作示例的更多信息，请参阅 对文档修订版执行编校。

因此，分类账保持了其整体数据的完整性，并通过现有验证 API 操作保持加密可验证性。您仍然可以按预期使用这些 API 操作来请求摘要 (GetDigest)、请求证明（GetBlock或 GetRevision），然后使用返回的对象运行验证算法。

重新计算修订哈希值

如果您计划通过重新计算其哈希来验证单个文档修订，您必须有条件地检查修订是否已被撤销。如果修订已被编辑，则可以使用dataHash字段中提供的哈希值。如果哈希值未被编辑，则可以使用该字段重新计算哈希。data

通过执行此条件检查，您可以识别已编辑的修订并采取适当的措施。例如，您可以记录数据操作事件以进行监视目的。

开始验证

在进行数据验证之前，您必须从您的分类帐请求一个摘要并保存它以备将来使用。在摘要覆盖的最新区块之前提交的任何文档修订都可以对该摘要进行验证。

然后，您向 Amazon QLDB 请求一个证明，以验证您想要验证的符合条件的修订。利用这个证明，您调用客户端 API 重新计算摘要，以您的修订哈希为起点。只要之前保存的摘要在 QLDB 之外是已知和受信任的，如果您重新计算的摘要哈希与保存的摘要哈希匹配，那么您的文档的完整性就得到了证明。

> **Important**
>
> - 您具体证明的是在您保存这个摘要和运行验证之间，文档修订没有被更改。您可以在稍后要验证的修订提交到日记账后立即请求并保存一个摘要。

- 作为最佳实践，我们建议您定期请求摘要并将其存储在离分类帐远离的地方。确定请求摘要的频率应基于您在分类帐中提交修订的频率。

  有关在现实用例背景下讨论加密验证价值的详细 AWS 博客文章，请参阅使用 Amazon QLDB 进行真实世界的加密验证。

有关如何从账本中请求摘要然后验证数据的 step-by-step 指南，请参阅以下内容：

- 步骤 1：在 QLDB 中请求摘要
- 步骤 2：在 QLDB 中验证您的数据

步骤 1：在 QLDB 中请求摘要

Amazon QLDB 提供了一个 API，用于请求涵盖分类账中日记账当前提示 的摘要。日记账提示指的是 QLDB 收到您的请求时的最近提交的数据块。您可以使用 AWS Management Console、S AWS DK 或 AWS Command Line Interface (AWS CLI) 来获取摘要。

主题

- AWS Management Console
- QLDB API

AWS Management Console

使用 QLDB 控制台，按照以下步骤来还原资源。

请求摘要（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 在导航窗格中，选择分类账。
3. 在分类账列表中，选择要申请摘要的分类账名称。
4. 选择“获取摘要”。获取摘要对话框显示以下摘要详细信息：

   - 摘要 - 您请求的摘要的 SHA-256 哈希值。
   - 摘要提示地址 - 您请求的摘要所涵盖的日记中的最新区块位置。地址包含以下两个字段：

  - `strandId` — 包含数据块的日记账链的唯一 ID。
  - `sequenceNo` — 一个索引号，用于指定数据块在链中的位置。
- 分类账 - 您请求摘要的分类账名称。
- 日期 - 您请求摘要时的时间戳。

5. 检查摘要信息。然后选择 Save（保存）。您可以保留默认文件名，或输入新名称。

> **Note**
>
> 您可能会注意到，即使您不修改分类账中的任何数据，您的摘要散列值和提示地址值也会发生变化。这是因为每次在 PartiQL 编辑器中运行查询时，控制台都会检索分类账的系统目录。这是向日记账提交的读事务，会导致最新区块地址发生变化。

此步骤将保存一个内容为 Amazon Ion 格式的纯文本文件。该文件的文件扩展名为 `.ion.txt`，包含前面对话框中列出的所有摘要信息。以下是摘要内容的示例。字段的顺序可能因您的浏览器而异。

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

6. 将这个文件保存在您将来可以访问的地方。随后，您可以使用这个文件来验证文档修订。

> **Important**
>
> 您稍后验证的文档修订必须包含在您保存的摘要中。也就是说，文档地址的序列号必须小于或等于摘要提示地址的序列号。

QLDB API

您还可以使用 Amazon QLDB API 与相关的 AWS SDK 或者使用 AWS CLI，通过向您的分类帐请求摘要。QLDB API 提供以下操作以供应用程序使用：

- GetDigest— 返回日记账中最新提交区块的账本摘要。响应包括一个 256 位的哈希值和一个块地址。

有关使用请求摘要的信息 AWS CLI，请参阅《命令参考》中的 get-diges t AWS CLI 命令。

示例应用程序

有关 Java 代码示例，请参阅 GitHub 存储库 a ws-samples/-amazon-qldb-dmv-sample java。有关如何下载和安装此示例应用程序的说明，请参阅 安装 Amazon QLDB Java 示例应用程序。在请求摘要之前，请确保按照 Java 教程 中的步骤 1-3 创建一个示例分类帐并用示例数据加载它。

课堂中的教程代码GetDigest提供了从`vehicle-registration`示例账本中请求摘要的示例。

要使用您保存的摘要验证文档修订，请继续执行 步骤 2：在 QLDB 中验证您的数据。

步骤 2：在 QLDB 中验证您的数据

Amazon QLDB 提供了一个 API，用于请求指定文档ID及其关联块的证明。您还必须提供之前保存的摘要的提示地址，如步骤 1：在 QLDB 中请求摘要中所述。您可以使用 AWS Management Console、S AWS DK 或 AWS CLI 来获取证据。

然后，您可以使用 QLDB 返回的证明通过客户端 API 验证文档修订与保存的摘要是否匹配。这使您能够控制用于验证数据的算法。

主题

- AWS Management Console
- QLDB API

AWS Management Console

本节描述了使用 Amazon QLDB 控制台验证文档修订与先前保存的摘要相匹配的步骤。

开始之前，请确保您已完成 步骤 1：在 QLDB 中请求摘要 中的步骤。验证需要先前保存的摘要，其中包含您要验证的修订。

验证文档修订（控制台）

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 首先，在分类账中查询要验证的文档修订的 `id`和 `blockAddress`。这些字段包含在文档的元数据中，您可以在提交视图中查询它们。

   文档 `id` 是系统分配的唯一 ID 字符串。`blockAddress` 是一种 Ion 结构，用于指定提交修订版本的区块位置。

在导航窗格中，选择 PartiQL 编辑器。

3. 选择要验证修订的分类帐名称。
4. 在查询编辑器中输入类似以下内容的语句，然后选择 SELECTRun Query（运行查询），或者按 。

```
SELECT metadata.id, blockAddress FROM _ql_committed_table_name
WHERE criteria
```

例如，以下查询从在 Amazon QLDB 控制台入门 中创建的示例分类帐中的 VehicleRegistration 表返回一个文档。

```
SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = 'KM8SRDHF6EU074761'
```

5. 复制并保存查询返回的 id 和 blockAddress 值。请务必省略 id 字段的双引号。在 Amazon Ion 中，字符串数据类型用双引号分隔。例如，您必须仅复制以下代码片段中的字母数字文本。

   "*LtMNJYNjSwzBLgf7sLifrG*"

6. 您已经选择了文档修订版，现在可以开始对其进行验证。

   在导航窗格中选择验证。

7. 在“验证文档”表单中，在“指定要验证的文档”下，输入以下输入参数：

   - 分类账-您要在其中验证修订的分类账。
   - 区块地址 - 您在步骤 4 中查询返回的blockAddress值。
   - 文档 ID - 您在步骤 4 中查询返回的 id 值。

8. 在“指定要验证的摘要”下，通过选择“选择摘要”，选择之前保存的摘要。如果文件有效，则会自动填充控制台上的所有摘要字段。或者，您可以直接从摘要文件中手动复制和粘贴以下值：

   - 摘要 - 摘要文件中的 digest 值。
   - 摘要提示地址 - 摘要文件中的 digestTipAddress 值。

9. 查看您的文档和摘要输入参数，然后选择Verify（验证）。

   控制台为您自动执行两个步骤：

   a. 向 QLDB 请求指定文档的证明。
   b. 使用 QLDB 返回的证明来调用客户端 API，该API会根据提供的摘要验证您的文档修订版本。要查看此验证算法，请参阅下一节QLDB API以下载代码示例。

控制台在验证结果卡中显示您的请求结果。有关更多信息，请参阅验证结果。

QLDB API

您还可以使用 Amazon QLDB API 与相关 AWS SDK 或者使用 AWS CLI，通过向 Amazon QLDB 请求验证文档修订。QLDB API 提供以下操作以供应用程序使用：

- `GetDigest` — 返回日记账中最新提交区块的分类账摘要。响应包括一个 256 位的哈希值和一个块地址。
- `GetBlock` — 返回日记账中指定地址的数据块对象。如果 `DigestTipAddress` 已提供，还会返回指定数据块的证明以供验证。
- `GetRevision` — 返回指定文档 ID 和块地址的修订数据对象。如果 `DigestTipAddress` 已提供，还将返回指定修订的用于验证的证明。

有关这些 API 操作的完整介绍，请参阅 Amazon QLDB API 参考。

有关使用验证数据的信息 AWS CLI，请参阅《AWS CLI 命令参考》。

示例应用程序

有关 Java 代码示例，请参阅 GitHub 存储库 a ws-samples/-amazon-qldb-dmv-sample java。有关如何下载和安装此示例应用程序的说明，请参阅 安装 Amazon QLDB Java 示例应用程序。在进行验证之前，请确保按照 Java 教程 中的步骤 1-3 创建一个示例分类帐并用示例数据加载它。

课堂中的教程代码GetRevision提供了一个示例，说明如何请求文档修订版的证据，然后验证该修订版。此类运行以下步骤：

1. 从示例分类账中请求新的摘要`vehicle-registration`。
2. 请求从 `vehicle-registration` 分类帐中的 `VehicleRegistration` 表中获取示例文档修订的证明。
3. 使用返回的摘要和证明验证示例修订。

验证结果

本节介绍上的 Amazon QLDB 数据验证请求返回的结果。 AWS Management Console有关如何提交验证请求的详细步骤，请参阅步骤 2：在 QLDB 中验证您的数据。

在 QLDB 控制台的验证页面上，您的请求结果显示在验证结果卡中。证明选项卡显示了 QLDB 为您指定的文档修订和摘要返回的证明的内容。其中包括以下内容：

- 修订哈希 — SHA-256 值，唯一表示您正在验证的文档修订。
- 证明哈希 — QLDB 提供的用于重新计算指定摘要的有序哈希列表。控制台从修订哈希开始，并依次与每个证明哈希组合，直到最终得到重新计算的摘要。

  默认情况下，列表处于折叠状态，因此您可以将其展开以显示哈希值。可选择按照 使用证明重新计算您的摘要 中描述的步骤自行尝试进行哈希计算。
- 计算的摘要 - 在修订哈希上进行的一系列哈希计算产生的哈希值。如果此值与您之前保存的摘要匹配，验证就成功了。

“区块”选项卡显示包含您正在验证的修订的块的内容。其中包括以下内容：

- 事务 ID — 提交数据块的事务唯一 ID。
- 事务时间 — 数据块提交到链的时间戳。
- 区块哈希 — 唯一表示此区块及其所有内容的 SHA-256 值。
- 区块地址 — 分类账日记账中提交区块的位置。地址包含以下两个字段：
  - Strand ID — 包含数据块的日记账链的唯一 ID。
  - 序列号 — 索引号，用于指定数据块在链中的位置。
- 语句 — 执行的 PartiQL 语句，用于提交该块中的条目。

> **Note**
>
> 如果以编程方式运行参数化语句，它们将以绑定参数的形式记录在您的日记账区块中，而不是文字数据。例如，您可能会在日记块中看到以下语句，其中问号（？）是文档内容的变量占位符。
>
> ```
> INSERT INTO Vehicle ?
> ```

- 文档条目 - 在该块中提交的文档修订。

如果您的请求未能验证文档修订，请参阅 常见的验证错误 以获取可能原因的信息。

使用证明重新计算您的摘要

在 QLDB 返回您的文档验证请求的证明后，您可以尝试自行进行哈希计算。本节概述了使用提供的证明重新计算摘要的高层步骤。

首先，将您的修订哈希与证明哈希列表中的第一个哈希配对。然后执行以下操作。

1. 对两个哈希进行排序。按照小端字节顺序比较这两个哈希的签名字节值。
2. 按照排序顺序连接这两个哈希。
3. 使用 SHA-256 哈希生成器对连接的对进行哈希处理。
4. 将您的新哈希与证明中的下一个哈希配对，并重复步骤 1-3。在您处理最后一个证明哈希之后，您的新哈希就是您重新计算的摘要。

如果您重新计算的摘要与之前保存的摘要相匹配，则说明您的文档已成功验证。

有关演示这些验证步骤的代码示例的 step-by-step 教程，请继续教程：使用 AWS 软件开发工具包验证数据。

教程：使用 AWS 软件开发工具包验证数据

在本教程中，您将通过软件开发工具包使用 QLDB API 来验证 Amazon QLDB 账本中的文档修订哈希和日志区块哈希。 AWS 您还可以使用 QLDB 驱动程序来查询文档修订。

举一个例子，您有一个文档修订，其中包含车辆识别码（VIN）为 `KM8SRDHF6EU074761` 的车辆的数据。文档修订位于一个名为 `vehicle-registration` 的分类帐中的一个名为 `VehicleRegistration` 的表中。假设您想要验证这辆车的文档修订以及包含修订的日记账区块的完整性。

> **Note**
>
> 有关在现实用例背景下讨论加密验证价值的详细 AWS 博客文章，请参阅使用 Amazon QLDB 进行真实世界的加密验证。

主题

- 先决条件
- 步骤 1：请求摘要
- 步骤 2：查询文档修订

先决条件

在开始之前，请务必执行以下操作：

1. 通过完成 Amazon QLDB 驱动程序入门 下的相应先决条件，为您选择的语言设置 QLDB 驱动程序。这包括注册 AWS、授予开发所需的编程访问权限以及配置您的开发环境。
2. 按照 Amazon QLDB 控制台入门 中的步骤 1-2，创建一个名为 `vehicle-registration` 的分类帐，并加载预定义的示例数据。

接下来，查看以下步骤以了解验证的工作原理，然后运行从头到尾的完整代码示例。

步骤 1：请求摘要

在您可以验证数据之前，您首先需要从您的分类帐`vehicle-registration`中请求一个摘要以供以后使用。

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')
```

步骤 2：查询文档修订

使用 QLDB 驱动程序查询与 VIN 相关联的块地址、哈希和文档 ID。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)
```

第 3 步：请求修订证明

遍历查询结果，对于每个块地址和文档 ID，以及分类帐名称，使用 QLDB 驱动程序提交一个 `GetRevision`请求。要获得修订的证明，您还必须提供先前保存的摘要中的提示地址。此 API 操作返回一个对象，其中包含文档修订和修订证明。

有关修订结构及其内容的信息，请参见查询文档元数据。

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)
```

然后，检索所请求修订的证明。

QLDB API 以节点哈希的有序列表的字符串表示形式返回证明。要将此字符串转换为节点哈希的二进制表示形式的列表，您可以使用 Amazon Ion 库中的 Ion 读取器。有关使用 Ion 库的更多信息，请参阅 Amazon Ion 说明书。

Java

在此示例中，您使用IonReader进行二进制转换。

```
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

在这个例子中，您使用 IonLoader 加载证明到一个 Ion 数据包中。

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

在此示例中，您使用 Ion 读取器将证明转换为二进制，并遍历证明的节点哈希列表。

```
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

在此示例中，您使用load函数进行二进制转换。

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

在此示例中，您使用loads函数进行二进制转换。

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

步骤 4：重新计算修订中的摘要

使用证明的哈希列表重新计算摘要，从修订哈希开始。只要之前保存的摘要在 QLDB 之外是已知和受信任的，如果重新计算的摘要哈希与保存的摘要哈希匹配，就证明了文档修订的完整性。

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))
```

第 5 步：申请日记账区块的证明

接下来，验证包含文档修订的日记账区块。

使用您在步骤 1 中保存的摘要中的屏蔽地址和提示地址提交GetBlock请求。与步骤 2 中的GetRevision请求类似，您必须再次提供已保存摘要中的提示地址才能获得区块的证明。此 API 操作返回一个包含区块和区块证明的对象。

有关日记账区块结构及其内容的信息，请参见Amazon QLDB 中的日记账内容。

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)
```

然后，从结果中检索区块哈希值和证明。

Java

在此示例中，您使用 IonLoader 将区块对象加载到 IonDatagram 容器中。

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

您还可以使用 IonLoader 将证明加载到 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

在此示例中，您使用 IonLoader 将区块和证明分别加载到离子数据报中。

```
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

在此示例中，您使用 Ion 读取器将证明转换为二进制，并遍历证明的节点哈希列表。

```
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

在此示例中，您使用`load`函数将区块和证明转换为二进制。

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

在此示例中，您使用`loads`函数将区块和证明转换为二进制。

```
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)
```

步骤 6：重新计算区块中的摘要

使用证明的哈希列表重新计算摘要，从区块哈希开始。只要之前保存的摘要在 QLDB 之外是已知和受信任的，如果重新计算的摘要哈希与保存的摘要哈希匹配，就证明了区块的完整性。

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))
```

前面的代码示例在重新计算摘要时使用以下dot函数。这个函数接受两个哈希作为输入，对它们进行排序，连接它们，然后返回连接数组的哈希值。

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
```

运行完整的代码示例

运行以下完整的代码示例，以执行从头到尾的所有先前的步骤。

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))
```

常见的验证错误

本节介绍了 Amazon QLDB 为验证请求引发的运行时错误。

以下是该服务返回的常见异常列表。每个异常包括具体的错误消息，接着是可能引发异常的 API 操作、一个简短的描述以及可能的解决方案建议。

IllegalArgumentException

消息：提供的 Ion 值无效，无法解析。

API 操作：`GetDigest, GetBlock, GetRevision`

在重试请求之前，请务必提供有效的 Amazon Ion 值。

IllegalArgumentException

消息：提供的块地址无效。

API 操作：`GetDigest, GetBlock, GetRevision`

在重试请求之前，请务必提供有效的块地址。地址是一种包含两个字段的 Amazon Ion 结构：即 `strandId` 和 `sequenceNo`。

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

IllegalArgumentException

消息：提供的摘要提示地址的序列号超出了该链路的最新提交记录。

API 操作：`GetDigest, GetBlock, GetRevision`

您提供的摘要提示地址的序号必须小于或等于日记链最新提交的记录的序号。在重试请求之前，请务必提供带有有效序号的摘要提示地址。

IllegalArgumentException

消息：提供的块地址的链路 ID 无效。

API 操作：`GetDigest, GetBlock, GetRevision`

您提供的块地址必须具有与该日记账的链 ID 相匹配的链 ID。在重试请求之前，请务必提供一个具有有效链路 ID 的块地址。

IllegalArgumentException

消息：提供的摘要区块的序列号超出了该组的最新提交记录。

API 操作：`GetBlock, GetRevision`

您提供的区块地址的序号必须小于或等于链最新提交的记录的序号。在重试请求之前，请务必提供带有有效序号的区块地址。

IllegalArgumentException

消息：提供的块地址的链 ID 必须与提供的摘要提示地址的链 ID 相匹配。

API 操作：`GetBlock, GetRevision`

只有当文档修订或块存在于与提供的摘要相同的日记账链中时，才能验证文档修订或块。

IllegalArgumentException

消息：提供的块地址的序列号不得大于所提供的摘要提示地址的序号。

API 操作：`GetBlock, GetRevision`

只有当您提供的摘要包含文档修订或块时，您才能验证它。这意味着它是在摘要提示地址之前提交给日记账的。

IllegalArgumentException

消息：在指定区块地址的区块中找不到提供的文档 ID。

API 操作：`GetRevision`

您提供的文档 ID 必须存在于您提供的块地址中。在重试请求之前，请确保这两个参数是一致的。

从Amazon QLDB 导出日记账数据

Amazon QLDB 使用不可变的事务日志（称为日记账）进行数据存储。日记账会跟踪已提交数据的每一次更改，并保存完整、可验证的更改历史记录。

您可以访问分类账中的日记账内容，以进行各种用途，包括分析、审计、数据保留、验证，以及导出到其他系统。以下主题描述了如何将您 AWS 账户中的日记账区块导出到您的 Amazon Simple Storage Service（Amazon S3）存储桶中。日记账导出作业将您的数据以 Amazon Ion 格式的文本或二进制表示，或 JSON Lines 文本格式的形式写入 Amazon S3 中作为对象。

在 JSON 行格式中，导出的数据对象中的每个区块都是由换行符分隔的有效 JSON 对象。您可以使用这种格式将 JSON 导出与 Amazon Athena 等分析工具直接集成，因为这些服务可以自动解析以换行 AWS Glue 符分隔的 JSON。有关数据格式的更多信息，请参阅 JSON 行。

有关 Amazon S3 的更多信息，请参阅 Amazon Simple Storage Service 用户指南。

> **Note**
>
> 如果您将 JSON 指定为导出作业的输出格式，QLDB 会将 Ion 日记账数据在导出的数据对象中将其转换为 JSON 格式。有关更多信息，请参阅向下转换至 JSON。

主题

- 请求在 QLDB 中导出日记账
- QLDB 中的日记账导出输出
- QLDB 中的日记账导出权限
- 日记账导出的常见错误

请求在 QLDB 中导出日记账

Amazon QLDB 提供了一个 API，用于请求导出指定日期和时间范围以及指定的 Amazon S3 存储桶目标的日记账区块。日记账导出作业可以以 Amazon Ion 格式的文本或二进制形式或 JSON 行文本格式写入数据对象。您可以使用 AWS Management Console、S AWS DK 或 AWS Command Line Interface (AWS CLI) 来创建导出任务。

主题

- AWS Management Console
- QLDB API
- 导出作业到期

AWS Management Console

请按照以下步骤使用 QLDB 控制台在 QLDB 中提交日记账导出请求。

请求导出（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 在导航窗格中，选择 导出。
3. 选择 创建导出作业。
4. 在创建导出作业页面上，输入以下导出设置：
   - 分类账 - 要导出其日记账区块的分类帐。
   - 开始日期和时间 - 要导出的日记账数据块范围的起始时间戳（协调世界时（UTC））。此时间戳必须早于结束日期和时间。如果您提供的开始时间戳早于分类账的 `CreationDateTime`，QLDB 会将其默认为分类账的 `CreationDateTime`。
   - 结束日期和时间 - 要导出的日记账区块范围的独占结束时间戳（UTC）。此日期和时间不能是未来的时间。
   - 日记账区块的目标 — 您的导出作业写入数据对象的 Amazon S3 存储桶和前缀名称。使用以下 Amazon S3 URI 格式。

     ```
     s3://DOC-EXAMPLE-BUCKET/prefix/
     ```

     您必须为输出对象指定 S3 存储桶名称和可选的前缀名称。以下是示例。

     ```
     s3://DOC-EXAMPLE-BUCKET/journalExport/
     ```

     存储桶名称和前缀都必须符合 Amazon S3 命名规则和惯例。有关命名存储桶更多信息，请参阅 Amazon Simple Storage Service 开发人员指南中的存储桶限制和约束。有关更多信息，请参阅对象键和元数据。

> **Note**
>
> 不支持跨区域查询。指定的 Amazon S3 存储桶必须与您的账本 AWS 区域 相同。

- S3 加密 - 您的导出作业在 Amazon S3 存储桶中写入数据时使用的加密设置。有关在 Amazon S3 中使用服务器端加密选项的信息，请参阅 Amazon S3 开发人员指南 中的使用服务器端加密保护数据。
  - 存储桶默认加密 - 使用指定 Amazon S3 存储桶的默认加密设置。
  - AES-256 - 具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）。
  - AWS-KMS — 使用带有 AWS KMS 托管密钥的服务器端加密 (SSE-KMS)。

    如果您选择此类型，并选择选择不同的 AWS KMS key某个选项，则您还必须在以下 Amazon 资源名称（ARN）格式中指定对称加密的 KMS 密钥。

    ```
    arn:aws:kms:aws-region:account-id:key/key-id
    ```

- 服务访问权限：在您的 Amazon S3 存储桶中授予 QLDB 写入权限的 IAM 角色。如果适用，IAM 角色还必须授予 QLDB 使用您的 KMS 密钥的权限。

  要在请求日记账导出时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 `iam:PassRole` 操作的权限。
  - 创建和使用新的服务角色 — 让控制台为您创建一个具有指定 Amazon S3 存储桶所需权限的新角色。
  - 使用现有的服务角色-要了解如何在 IAM 中手动创建此角色，请参阅导出权限。
- 输出格式 - 导出的日记账数据的输出格式
  - Ion 文本 - （默认）Amazon Ion 的文本表示形式
  - Ion 二进制 — Amazon Ion 的二进制表示
  - JSON - 以换行符分隔的 JSON 文本格式

    如果您选择 JSON，QLDB 会在导出的数据对象中将 Ion 日记账数据向下转换为 JSON。有关更多信息，请参阅向下转换至 JSON。

5. 根据需要进行设置后，选择 创建导出作业。

   完成导出作业所需的时间取决于数据大小。如果您的请求提交成功，控制台将返回主导出页面，并列出您的导出作业及其当前状态。
6. 您可以在 Amazon S3 控制台上查看导出对象。

通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

要了解有关这些输出对象格式的更多信息，请参阅 QLDB 中的日记账导出输出。

> **Note**
>
> 导出作业将在完成七天后过期。有关更多信息，请参阅导出作业到期。

QLDB API

您也可以使用带软件开发工具包的 Amazon QLDB API 或者，请求日记导出。 AWS AWS CLI QLDB API 提供以下操作以供应用程序使用：

- `ExportJournalToS3`— 将日期和时间范围内的日记账内容从给定分类账导出到指定的 Amazon S3 存储桶。导出作业可以 Amazon Ion 格式的文本或二进制形式或 JSON 行文本格式写入数据对象。
- `DescribeJournalS3Export`— 返回有关日记账导出作业的详细信息。输出包括其当前状态、创建时间和原始导出请求的参数。
- `ListJournalS3Exports`— 返回与当前 AWS 账户 和地区相关联的所有账本的日记账导出作业描述的列表。每个导出作业描述的输出都包含与 `DescribeJournalS3Export` 返回的相同的详细信息。
- `ListJournalS3ExportsForLedger`— 返回给定分类账的日记账导出作业描述列表。每个导出作业描述的输出都包含与 `DescribeJournalS3Export` 返回的相同的详细信息。

有关这些 API 操作的完整介绍，请参阅 Amazon QLDB API 参考。

有关使用导出日记账数据的信息 AWS CLI，请参阅AWS CLI 命令参考。

Java 中的示例应用程序

有关基本导出操作的 Java 代码示例，请参阅 GitHub 存储库 a ws-samples/-amazon-qldb-dmv-sample java。有关如何下载和安装此示例应用程序的说明，请参阅 安装 Amazon QLDB Java 示例应用程序。在请求导出之前，请确保按照 Java 教程 中的步骤 1-3 创建一个示例分类帐并用示例数据加载它。

以下类中的教程代码提供了创建导出、检查导出状态和处理导出输出的示例。

类	描述
ExportJournal	从vehicle-registration 示例分类账中导出日记账区块，时间戳范围为 10 分钟前到现在。将输出对象写入指定的 S3 存储桶，如果未提供唯一存储桶，则创建唯一存储桶。
DescribeJournalExport	描述vehicle-registration 示例分类账exportId中指定的日记账导出作业。
ListJournalExports	返回vehicle-registration 示例分类账的日记账导出作业描述列表。
ValidateQldbHashChain	使用给定的exportId验证vehicle-registration 示例分类账的哈希链。如果未提供，则请求新的导出以用于哈希链验证。

导出作业到期

已完成的日记账导出作业有 7 天的保留期。此限制到期后，它们会自动被硬删除。此到期时间是一项硬性限制，无法更改。

在删除已完成的导出作业后，您将无法再使用 QLDB 控制台或以下 API 操作来检索有关该任务的元数据：

- DescribeJournalS3Export
- ListJournalS3Exports
- ListJournalS3ExportsForLedger

但是，此过期时间对导出数据本身没有影响。所有元数据都保留在导出文件中写入的清单文件中。此过期时间旨在提供更顺畅的API操作体验，以列出日记账导出作业。QLDB 会移除旧的导出作业，确保您只看到最近的导出，而不必解析多页作业。

QLDB 中的日记账导出输出

除包含您的日记账区块的数据对象外，Amazon QLDB 日记账导出作业还会写入两个清单文件。这些文件都保存在导出请求指定的 Amazon S3 存储桶中。以下章节介绍每个输出对象的格式和内容。

> **Note**
>
> 如果您将 JSON 指定为导出作业的输出格式，QLDB 会将 Amazon Ion 日记账数据在导出的数据对象中将其转换为 JSON 格式。有关更多信息，请转至 向下转换至 JSON。

主题

清单文件

Amazon QLDB 为每个导出请求在提供的 S3 存储桶中创建两个清单文件。初始清单文件将在您提交导出请求后立即创建。最终清单文件是在导出完成后写入的。您可以使用这些文件来检查 Amazon S3 中导出作业的状态。

清单文件内容的格式与请求的导出输出格式相对应。

初始清单

初始清单表明您的导出作业已启动。它包含您传递给请求的输入参数。除了 Amazon S3 目标以及导出的开始和结束时间参数外，此文件还包含一个 `exportId`。此 `exportId` 是 QLDB 分配给每个导出作业的唯一 ID。

文件命名约定如下。

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

下面是 Ion 文本格式的初始清单文件及其内容示例。

```
s3://DOC-EXAMPLE-BUCKET/journalExport/8UyXulxccYLAsbN1aon7e4.started.manifest
```

```
{
  ledgerName:"my-example-ledger",
  exportId:"8UyXulxccYLAsbN1aon7e4",
```

```
  inclusiveStartTime:2019-04-15T00:00:00.000Z,
  exclusiveEndTime:2019-04-15T22:00:00.000Z,
  bucket:"DOC-EXAMPLE-BUCKET",
  prefix:"journalExport",
  objectEncryptionType:"NO_ENCRYPTION",
  outputFormat:"ION_TEXT"
}
```

`outputFormat`只有在导出请求中指定时，初始清单才会包含该清单。如果您没有指定输出格式，导出的数据将默认为 `ION_TEXT`。

DescribeJournalS3Export API 操作和导出的 Amazon S3 对象的内容类型也表明了输出格式。

最终清单

最终清单表示特定日记账链的导出作业已完成。导出作业为每个分支写入一个单独的最终清单文件。

> **Note**
>
> 在 Amazon QLDB 中，分支是分类账日记账的分区。QLDB 目前仅支持单链日记账。

最终清单包括导出期间写入的数据对象密钥的有序列表。文件命名约定如下。

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

`strandId` 是 QLDB 分配给链的唯一 ID。下面是 Ion 文本格式的最终清单文件及其内容示例。

```
s3://DOC-EXAMPLE-BUCKET/
journalExport/8UyXulxccYLAsbN1aon7e4.JdxjkR9bSYB5jMHWcI464T.completed.manifest
```

```
{
  keys:[
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-4.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.5-10.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.11-12.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.13-20.ion",
    "2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.21-21.ion"
  ]
}
```

数据对象

Amazon QLDB 会以 Amazon Ion 格式的文本或二进制表示，或 JSON Lines 文本格式，将日记账数据对象写入提供的 Amazon S3 存储桶中。

在 JSON 行格式中，导出的数据对象中的每个区块都是由换行符分隔的有效 JSON 对象。您可以使用这种格式将 JSON 导出与 Amazon Athena 等分析工具直接集成，因为这些服务可以自动解析以换行 AWS Glue 符分隔的 JSON。有关数据格式的更多信息，请参阅 JSON 行。

数据对象名称

日记账导出作业使用以下命名约定写入这些数据对象。

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- 每个导出作业的输出数据都被分成多个块。
- `yyyy/mm/dd/hh`— 您提交导出请求的日期和时间。在同一小时内导出的对象是按相同的 Amazon S3 前缀进行分组。
- `strandId`— 包含要导出的日记账区块的特定链的唯一 ID。
- `startSn-endSn`— 对象中包含的序列号范围。序列号指定区块在链中的位置。

例如，假设您指定以下路径。

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

您的导出作业会创建一个类似于以下的 Amazon S3 数据对象。此示例显示了 Ion 格式的对象名称。

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

数据对象内容

每个数据对象都包含以下格式的日记账区块对象。

```
{
  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
    }
  ]
}
```

区块是在事务过程中提交到日记账的对象。区块包含事务元数据、以及代表事务中提交的文档修订版本条目、和提交这些修订的 PartiQL 语句。

以下是一个带有 Ion 文本格式样本数据的模块示例。有关块对象字段的更多信息，请参阅Amazon QLDB 中的日记账内容。

> **Note**
>
> 数据块示例仅用于参考。显示的哈希值并非实际计算的哈希值。

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

在revisions字段中，某些修订版本对象可能只包含一个 hash 值而不包含其他属性。这是仅供内部使用的系统修订版，不包含用户数据。导出作业将这些修订包含在各自的区块中，因为这些修订的哈希值是日记完整哈希链的一部分。加密验证需要完整的哈希链。

向下转换至 JSON

如果您将 JSON 指定为导出作业的输出格式，QLDB 会将 Amazon Ion 日记账数据在导出的数据对象中将其转换为 JSON 格式。但是，在某些情况下，如果您的数据使用 JSON 中不存在的丰富 Ion 类型，则将 Ion 转换为 JSON 时会有损失。

有关 Ion 与 JSON 转换规则的详细信息，请参阅Amazon Ion Cookbook中的向下转换至 JSON。

导出处理器库（Java）

QLDB 为 Java 提供了一个可扩展的框架，该框架可以简化 Amazon S3 中的导出处理。该框架库处理读取导出的输出并按顺序遍历导出的区块的工作。要使用此导出处理器，请参阅 GitHub 存储库 awslabs/-java amazon-qldb-export-processor。

QLDB 中的日记账导出权限

在向 Amazon QLDB 提交日记账导出请求之前，您必须在指定的 Amazon S3 存储桶中为 QLDB 提供写入权限。如果您选择将客户托管 AWS KMS key 的作为 Amazon S3 存储桶的对象加密类型，您还必须授予 QLDB 使用您指定的对称加密密钥的权限。Amazon RDS 不支持非对称 KMS 密钥。

要为您的导出作业提供必要的权限，您可以让 QLDB 扮演具有相应权限策略的 IAM 服务角色。服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务委派权限的角色。

> **Note**
>
> 要在请求日记账导出时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 `iam:PassRole` 操作的权限。这是对 QLDB 分类账资源的 `qldb:ExportJournalToS3` 权限的补充。
>
> 要了解如何使用 IAM 控制对 QLDB 的访问权限，请参阅 Amazon MQ 如何与 IAM 协同工作。有关 QLDB 策略示例，请参阅Amazon QLDB 基于身份的策略示例。

在此示例中，您创建一个角色，允许 QLDB 代表您将对象写入 Amazon S3 存储桶。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务委派权限的角色。

如果您是首次在中 AWS 账户 导出 QLDB 日记，则必须先通过执行以下操作创建具有相应策略的 IAM 角色。或者，您可以使用 QLDB 控制台自动为您创建角色。否则，您可以选择之前创建的角色。

主题

- 创建权限策略
- 创建 IAM 角色

创建权限策略

完成以下步骤来为 QLDB 日记账导出作业创建权限策略。此示例显示了 Amazon S3 存储桶策略，该策略授予 QLDB 向您的指定存储桶写入对象的权限。如果适用，示例还显示了一个密钥策略，允许 QLDB 使用您指定的对称加密 KMS 密钥。

有关 Amazon S3 的存储桶策略的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的使用存储桶策略和用户策略。有关 AWS KMS 密钥策略的更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用 AWS KMS中的密钥策略。

> **Note**
>
> 您的 Amazon S3 存储桶和 KMS 密钥必须与您的 QLDB 账本 AWS 区域 相同。

使用 JSON 策略编辑器创建策略

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在左侧的导航栏中，选择 Policies（策略）。

   如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择开始使用。
3. 在页面的顶部，选择 Create Policy（创建策略）。
4. 请选择 JSON 选项卡。
5. 输入 JSON 策略文档。
   - 如果您正在使用客户托管的 KMS 密钥进行 Amazon S3 对象加密，请使用以下示例策略文档。*############ DOC-EXAMPLE-BUCKET#us-east-1#123456789012 # 1234abcd- 12ab-34cd-56ef-12345678 90ab ##########*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        },
        {
            "Sid": "QLDBJournalExportKMSPermission",
            "Action": [ "kms:GenerateDataKey" ],
            "Effect": "Allow",
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

- 对于其他加密类型，请使用以下示例策略文档。要使用此策略，请将*#### DOC-EXAMPLE-BUCKET ####### Amazon S3 ###*名称。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. 选择Review policy（查看策略）。

> **Note**
>
> 您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy（查看策略），IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在 Review policy（查看策略）页面上，为创建的策略输入 Name（名称）和 Description（说明）（可选）。查看策略摘要以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。

创建 IAM 角色

为您的 QLDB 日记账导出作业创建权限策略后，您可以创建一个 IAM 角色并将您的策略附加到该角色。

创建用于 QLDB 的服务角色（IAM 控制台）

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，请选择 QLDB，然后选择 QLDB 使用案例。
5. 选择下一步。
6. 选中您之前创建的策略旁边的方框。
7. （可选）设置权限边界。这是一项高级特征，可用于服务角色，但不可用于服务相关角色。

   a. 打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。

      IAM 包含您账户中的 AWS 托管策略和客户托管策略列表。

   b. 选择要用于权限边界的策略。
8. 选择下一步。
9. 输入有助于识别角色的作用的角色名称或者角色名称后缀。

   > **Important**
   >
   > 命名角色时，请注意以下事项：
   >
   > - 角色名称在您内部必须是唯一的 AWS 账户，并且不能因大小写而变得唯一。
   >
   >   例如，不要同时创建名为 **PRODROLE** 和 **prodrole** 的角色。当角色名称在策略中使用或者作为 ARN 的一部分时，角色名称区分大小写，但是当角色名称在控制台中向客户显示时（例如，在登录期间），角色名称不区分大小写。
   > - 创建角色后，您无法编辑该角色的名称，因为其他实体可能会引用该角色。
10. （可选）对于描述，输入角色的描述。
11. （可选）要编辑角色的使用案例和权限，请在步骤 1：选择可信实体或步骤 2：添加权限部分中选择编辑。

12. （可选）为了帮助识别、组织或搜索角色，请以键值对形式添加标签。有关在 IAM 中使用标签的更多信息，请参阅《IAM 用户指南》中的标记 IAM 资源。
13. 检查该角色，然后选择创建角色。

以下的 JSON 文档是一个信任策略的示例，该策略允许 QLDB 假定一个附有特定权限的 IAM 角色。

```
{
    "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**
>
> 以下示例演示如何使用 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键来防范混淆代理问题。通过这个信任策略，QLDB 只能为同一账户123456789012中的任何 QLDB 资源扮演角色。
> 有关更多信息，请参阅防止跨服务混淆座席。

创建 IAM 角色后，返回 QLDB 控制台并刷新“创建导出作业”页面，以便它可以找到您的新角色。

日记账导出的常见错误

本节介绍了 Amazon QLDB 为日记账导出请求引发的运行时系统错误。

以下是该服务返回的常见异常列表。每个异常都包括特定的错误消息，然后是简短的描述和可能的解决方案建议。

AccessDeniedException

######UserArn #####iam#PassRole #####ro Learn

您无权将 IAM 角色传递给 QLDB 服务。QLDB 要求所有日记账导出请求都有一个角色，并且您必须具有将该角色传递给 QLDB 的权限。该角色为 QLDB 提供了在您指定的 Amazon S3 存储桶中的写入权限。

核实您定义了一个 IAM policy，该策略授予对您指定的 QLDB 服务（`qldb.amazonaws.com`）的 IAM 角色资源执行 `PassRole` API 操作的权限。有关策略示例，请参阅Amazon QLDB 基于身份的策略示例。

IllegalArgumentException

消息：QLDB 在验证 S3 配置时遇到错误：*errorCode errorMessage*

导致此错误的一个可能原因是 Amazon S3 中不存在所提供的 Amazon S3 存储桶。或者，QLDB 没有足够的权限将对象写入您指定的 Amazon S3 存储桶。

验证您在导出任务请求中提供的 S3 存储桶名称是否正确。有关命名存储桶的信息，请参阅Amazon Simple Storage Service 户指南 中的 存储桶限制和局限。

此外，请确认您是否为指定的存储桶定义了一个向 QLDB 服务（`qldb.amazonaws.com`）授予 `PutObject` 和 `PutObjectAcl` 权限的策略。要了解更多信息，请参阅 导出权限。

IllegalArgumentException

消息：在验证 S3 配置时，来自 Amazon S3 的意外响应。来自 S3 的响应：*errorCode errorMessage*

尝试将日记账导出数据写入提供的 S3 失败，并显示提供的 Amazon S3 错误响应。有关可能原因的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的Amazon S3 故障排除。

IllegalArgumentException

消息：Amazon S3 存储桶前缀不得超过 128 个字符

日记账导出请求中提供的前缀包含超过 128 个字符。

IllegalArgumentException

消息：开始日期不得早于结束日期

`InclusiveStartTime` 和 `ExclusiveEndTime` 必须采用 ISO 8601 日期和时间格式以及通用协调时间（UTC）。

IllegalArgumentException

消息：结束日期不能是未来的日期

`InclusiveStartTime` 和 `ExclusiveEndTime` 都必须采用 `ISO 8601` 日期和时间格式以及 UTC。

IllegalArgumentException

消息：提供的对象加密设置 (S3EncryptionConfiguration) 与 AWS Key Management Service (AWS KMS) 密钥不兼容

您 `KMSKeyArn` 提供的是 `NO_ENCRYPTION` 或 `ObjectEncryptionType` 中的一个 `SSE_S3`。您只能为客户提供对象加密类型的托管 AWS KMS key `SSE_KMS`。有关在 Amazon S3 中使用服务器端加密选项的信息，请参阅 Amazon S3 开发人员指南 中的使用服务器端加密保护数据。

LimitExceededException

消息：已超出 2 个同时运行日记账导出任务的限制

QLDB 强制执行两个并发日记账导出任务的默认限制。

Amazon QLDB 流式传输日记账数据

Amazon QLDB 使用不可变的事务日志（称为日记账）进行数据存储。日记账会跟踪已提交数据的每一次更改，并保存完整、可验证的更改历史记录。

您可以在 QLDB 中创建一个流，其捕获提交到您日记账的每个文档修订版本，并将此数据近实时传送到 Amazon Kinesis Data Streams。QLDB 流是从分类账的日记账到 Kinesis Data Streams 资源的连续数据流。

然后，您使用 Kinesis 流平台或 Kinesis Client Library 来使用流、处理数据记录和分析数据内容。QLDB 流通过三种类型的记录将您的数据写入 Kinesis Data Streams：控件、区块摘要和修订详情。有关更多信息，请参阅QLDB 在 Kinesis 中流记录。

主题

常见使用案例

流式传输使您能够将 QLDB 作为单一、可验证的真实来源，并将日记账数据与其他服务集成。以下是 QLDB 日记账流支持的一些常见用例：

- 事件驱动架构 - 使用解耦组件以事件驱动的架构风格构建应用程序。例如，银行可以使用 AWS Lambda 功能来实现通知系统，当客户的账户余额降至阈值以下时，该系统会提醒客户。在这样的系统中，账户余额保存在 QLDB 分类账中，任何余额变化都记录在日记账中。该 AWS Lambda 函数可以在消耗提交到日志并发送到 Kinesis 数据流的余额更新事件时触发通知逻辑。

- 实时分析 - 构建 Kinesis 使用者应用程序，对事件数据进行实时分析。借助此功能，您可以近乎实时地获得见解，并对不断变化的业务环境做出快速响应。例如，电子商务网站可以分析产品销售数据，并在销售达到限制时立即停止打折产品的广告。
- 历史分析 — 通过重播历史事件数据，利用 Amazon QLDB 面向日记账的架构。您可以选择从过去任何时间点开始 QLDB 流，在该流中，自那时以来的所有修订都将传输到 Kinesis Data Streams。使用此功能，您可以构建 Kinesis 使用者应用程序，对历史数据运行分析作业。例如，电子商务网站可以根据需要运行分析，从而生成以前未捕获的过去销售指标。
- 复制到专用数据库 — 使用 QLDB 日记账流将 QLDB 分类账连接到其他专门构建的数据存储。例如，使用 Kinesis 流数据平台与亚马逊 OpenSearch 服务集成，后者可以为 QLDB 文档提供全文搜索功能。您还可以构建自定义 Kinesis 使用者应用程序，将您的日记账数据复制到其他提供不同实体化视图的专用数据库。例如，将关系数据复制到 Amazon Aurora，或者将基于图表的数据复制到 Amazon Neptune。

消耗您的流

使用 Kinesis Data Streams 持续使用、处理和分析大量数据记录。除了 Kinesis Data Streams 之外，Kinesis 流数据平台还包括亚马逊数据 Firehose 和适用于 Apach e Flink 的亚马逊托管服务。您可以使用此平台将数据记录直接发送到亚马逊服务、Amazon Redshift、Amazon S3 或 Splunk 等 OpenSearch 服务。有关更多信息，请参阅 Amazon Kinesis 数据流开发人员指南 中的 Amazon Kinesis Data Streams 消费者。

您还可以使用 Kinesis 客户端库（KCL）构建流使用者应用程序，从而以自定义方式处理数据记录。KCL 提供低级 Kinesis Data Streams API 之上的有用抽象来简化编码。要了解有关 KCL 的详细信息，请参阅 Amazon Kinesis Data Streams 开发人员指南的使用 Kinesis 客户端库。

交付保证

QLDB 直播提供了at-least-once传输保证。由 QLDB 流生成的每条数据记录至少会传输到 Kinesis Data Streams 一次。相同的记录可以多次出现在 Kinesis 数据流中。因此，如果您的使用案例需要，您必须在消费应用程序层中实现去重逻辑。

也没有订购保证。在某些情况下，可能会在 Kinesis 数据流中乱序生成 QLDB 区块和修订。有关更多信息，请参阅处理重复和 out-of-order记录。

传送延迟注意事项

QLDB流通常以接近实时的方式将更新传递到 Kinesis Data Streams。但是，在将新提交的 QLDB 数据发送到 Kinesis 数据流之前，以下情况可能会造成其他的延迟：

- Kinesis 可以限制从 QLDB 流式传输的数据，具体取决于您的 Kinesis Data Streams 配置如何。例如，如果您有多个 QLDB 流写入单个 Kinesis 数据流，而 QLDB 的请求速率超过 Kinesis 流资源的容量，则可能发生这种情况。在使用按需预配时，如果吞吐量在不到 15 分钟内增长到超过前一个峰值的两倍以上，Kinesis 中也可能发生节流。

  您可以通过监控 Kinesis 指标 `WriteProvisionedThroughputExceeded` 来衡量超过的吞吐量。有关更多信息和可能的解决方案，请参阅如何排查 Kinesis Data Streams 中的节流错误问题？。
- 使用 QLDB 流，您可以创建一个不确定的流，其开始日期和时间为过去且没有结束日期和时间。根据设计，只有在成功交付指定开始日期和时间的所有先前数据之后，QLDB 才开始向 Kinesis Data Streams 发送新提交的数据。如果您在这种情况下感觉到额外的延迟，您可能需要等待先前的数据被传递，或者您可以从较晚的开始日期和时间开始流。

数据流入门

以下简要概述了开始将日记账数据流式传输到 Kinesis Data Streams 所需的步骤：

1. 创建 Kinesis Data Streams 资源。有关说明，请参阅 Amazon Kinesis Data Streams 开发者指南中的创建和更新数据流。
2. 创建一个 IAM 角色，允许 QLDB 承担对Kinesis 数据流进行写入操作的权限。有关说明，请参阅 QLDB 中的流权限。
3. 创建 QLDB 日记账流。有关说明，请参阅 在 QLDB 中创建和管理流。
4. 如上一节消耗您的流所述，使用 Kinesis 数据流。有关演示如何使用 Kinesis 客户端库或的代码示例 AWS Lambda，请参阅。在 QLDB 中使用流进行开发

在 QLDB 中创建和管理流

Amazon QLDB 提供 API 操作，用于创建和管理从您的分类账到 Amazon Kinesis Data Streams 的日记账数据流。QLDB 流捕获提交到日记账的每个文档修订，然后将其发送到 Kinesis 数据流。

您可以使用 AWS Management Console、S AWS DK 或 AWS Command Line Interface (AWS CLI) 来创建日记流。此外，您还可以使用 AWS CloudFormation 模板来创建流。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的AWS::QLDB::Stream资源。

主题

流参数

要创建 QLDB 日记账流，您必须提供以下配置参数：

分类账

您想将其日记账数据流式传输到 Kinesis Data Streams 的 QLDB 分类账。

流名称

要分配给 QLDB 日记账流的名称。用户定义的名称有助于识别和指示流的用途。

您的流名称在给定分类账的其他活动的 流中必须是唯一的。流名称与分类账名称具有相同的命名约束，如 Amazon QLDB 资源中的限额和限制 中所定义。

除了流名称外，QLDB 还会为您创建的每个 QLDB 流分配一个流 ID。在给定分类账的所有流中，流 ID 是唯一的，无论其状态如何。

开始日期和时间

开始流式传输日记账数据的开始日期和时间。此值可以是过去的任何日期和时间，但是不能是将来的任何日期和时间。

结束日期和时间

（可选）指定流结束的日期和时间。

如果您创建了一个没有结束时间的无限期流，您必须手动取消它以结束流。您还可以取消一个尚未达到指定结束日期和时间的活动有限流。

目标 Kinesis 数据流

您的数据流写入数据记录的 Kinesis Data Streams 目标资源。要了解如何创建 Kinesis 数据流，请参阅Amazon Kinesis Data Streams 开发者指南 中的 创建和更新数据流。

> **Important**
>
> - 不支持跨账户和跨区域流。指定的Kinesis数据流必须与您的账本位于相同的 AWS 区域和分类账中。
> - 默认情况下，Kinesis Data Streams 中的记录聚合处于启用状态。此选项使 QLDB 能够在单个 Kinesis 数据流记录中发布多个数据记录，从而增加每个 API 调用发送的记录数量。
>
>   记录聚合对记录处理具有重要影响，并且需要在流使用者中取消聚合。要了解更多信息，请参阅 《Amazon Kinesis Data Streams 开发人员指南》 中的 KPL 主要概念以及使用者取消聚合。

IAM 角色

允许 QLDB 承担您的 Kinesis 数据流的写入权限的 IAM 角色。您可以使用 QLDB 控制台自动创建此角色，也可以在 IAM 中手动创建该角色。要了解如何手动创建，请参阅流权限。

要在请求日记账流时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 `iam:PassRole` 操作的权限。

流 ARN

每个 QLDB 日记账流都是分类账的子资源，由 Amazon 资源名称（ARN）进行唯一标识。以下是名为 `IiPT4brpZCqCq3f4MTHbYy` 分类账而且流 ID 为 `exampleLedger` 的 QLDB 流的示例 ARN。

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

以下部分介绍如何使用 AWS Management Console创建和取消 QLDB 流式传输。

AWS Management Console

按照以下步骤，使用 QLDB 控制台创建或取消 QLDB 流。

创建流（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 在导航窗格中，选择 Streams（流）。
3. 选择 创建 QLDB 流。
4. 在 创建 QLDB 流页面，输入以下设置：

   - 流名称 - 要分配给 QLDB 日记账流的名称。
   - 分类账 — 要流式传输其日记账数据的分类账。
   - 开始日期和时间：开始流式传输日记账数据的开始时间戳（UTC）。此时间戳默认为当前日期和时间。时间不能在将来中，并且必须早于结束日期和时间。
   - 结束日期和时间-（可选）指定流结束时间的互斥的时间戳（UTC）。如果不定义此参数，则流将无限期运行，直到您取消它。
   - 目标流 — 您的数据流向其写入数据记录的 Kinesis Data Streams 目标资源。采用以下 ARN 格式。

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     以下是示例。

     ```
     arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
     ```

     不支持跨账户和跨区域流。指定的 Kinesis 数据流必须 AWS 区域 与您的账本相同。
   - 在 Kinesis Data Streams 中启用记录聚合-（默认启用）允许 QLDB 在单个 Kinesis Data Streams 记录中发布多个数据记录，从而增加每个 API 调用发送的记录数量。
   - 服务访问权限 — 授予 QLDB 对您的 Kinesis 数据流的写入权限的 IAM 角色。

     要在请求日记账流时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 *iam:PassRole* 操作的权限。

     - 创建和使用新的服务角色 - 让控制台为您创建一个具有指定 Kinesis 数据流所需权限的新角色。
     - 使用现有的服务角色-要了解如何在 IAM 中手动创建此角色，请参阅流权限。
   - 标签 –（可选）通过以键值对的形式附加标签来向角色添加元数据。您可以向流中添加标签来帮助组织和标识这些它们。有关更多信息，请参阅为 Amazon QLDB 资源贴标签。

选择添加标签，然后根据需要输入任何键值对。

5. 根据需要进行设置后，选择 创建 QLDB 流。

   如果您的请求提交成功，控制台将返回主流页面，并列出您的 QLDB 流及其当前状态。

6. 在您的流处于活动状态之后，使用 Kinesis 来使用一个使用者应用程序处理您的流数据。

   打开 Kinesis 数据流控制台，网址为：https://console.aws.amazon.com/kinesis。

有关流中数据记录格式的信息，请参阅 QLDB 在 Kinesis 中流记录。

要了解如何处理导致错误的流，请参阅处理受损流。

取消流（控制台）

取消 QLDB 流后，您将无法重启该流。要恢复向 Kinesis Data Streams 传输数据，您可以创建一个新的 QLDB 流。

1. 通过以下网址打开 Amazon QLDB 控制台：https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择 Streams（流）。
3. 在 QLDB 流列表中，选择要取消的当前流流。
4. 选择 取消流。请在提供的框中输入 **`cancel stream`** 以确认。

有关将 QLDB API 与 SDK 配合 AWS 使用来创建和管理日记流的信息，请参阅。 AWS CLI 在 QLDB 中使用流进行开发

流状态

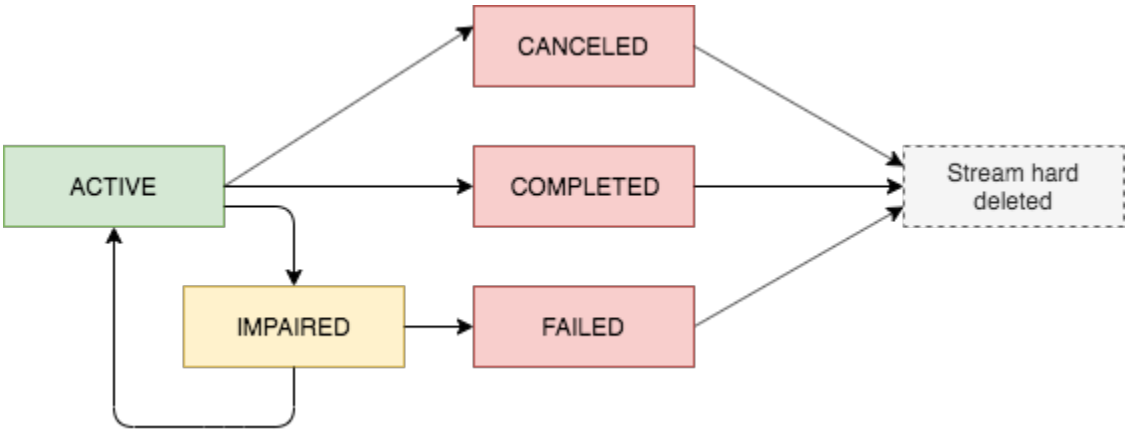
QLDB 流可能处于以下状态之一：

- `ACTIVE`— 当前正在流式传输或等待流式传输数据（对于没有结束时间的无限期流）。
- `COMPLETED`— 已成功完成指定时间范围内所有日记账区块的流式传输。这是最终状态。
- `CANCELED`— 在指定的结束时间之前由用户请求结束，并且不再主动流式传输数据。这是最终状态。
- `IMPAIRED`— 由于出现需要您采取措施的错误，无法向 Kinesis 写入记录。这是一种可恢复的非终止状态。

如果您在一小时内解决了错误，则流会自动进入 ACTIVE 状态。如果错误在一小时后仍未解决，则流会自动进入 FAILED 状态。

- FAILED— 由于出错，无法向 Kinesis 写入记录，并且处于无法恢复的终止状态。

下图说明了 QLDB 流资源如何在状态之间转换。



终端流到期

处于终端状态（CANCELED、COMPLETED、和FAILED）的流资源的保留期为 7 天。此限制到期后，它们会自动被硬删除。

删除终端流后，您将无法再使用 QLDB 控制台或 QLDB API 来描述或列出流资源。

处理受损流

如果您的流遇到错误，它会先移至 IMPAIRED 状态。QLDB 会继续 IMPAIRED重 试流长达一小时。

如果您在一小时内解决了错误，则流会自动进入 ACTIVE 状态。如果错误在一小时后仍未解决，则流会自动进入 FAILED 状态。

受损或失败的流可能由以下错误原因之一：

- KINESIS_STREAM_NOT_FOUND— 目标 Kinesis Data Streams 资源不存在。验证您在 QLDB 流请求中提供的 Kinesis 数据流是否正确。然后，前往 Kinesis 并创建您指定的数据流。
- IAM_PERMISSION_REVOKED— QLDB 没有足够的权限将数据记录写入您指定的 Kinesis 数据流。请确认您是否为指定的 Kinesis 数据流定义了一个策略，该策略授予 QLDB 服务（qldb.amazonaws.com）对以下操作的权限：

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

监控受损流

如果流受损，QLDB 控制台会显示一条横幅，显示有关该流及其遇到的错误的详细信息。您还可以使用 `DescribeJournalKinesisStream` API 操作来获取流的状态和潜在的错误原因。

此外，您还可以使用 Amazon CloudWatch 创建用于监控直播`IsImpaired`指标的警报。有关使用监控 QLDB 指标 CloudWatch的信息，请参阅。Amazon QLDB 指标与维度

在 QLDB 中使用流进行开发

本节总结了您可以与 AWS 软件开发工具包一起使用的 API 操作，或者在 Amazon QLDB 中创建和管理日记流。 AWS CLI 它还描述了演示这些操作并使用 Kinesis 客户端库（KCL）或 AWS Lambda 实现流使用者的示例应用程序。

可以使用 KCL 为 Amazon Kinesis Data Streams 构建用户应用程序。KCL 提供低级 Kinesis Data Streams API 之上的有用抽象来简化编码。要了解有关 KCL 的详细信息，请参阅 Amazon Kinesis Data Streams 开发人员指南的使用 Kinesis 客户端库。

目录

QLDB 日记账流 API

QLDB API 提供以下日记账流操作以供应用程序使用：

- `StreamJournalToKinesis`— 为给定的 QLDB 分类账创建日记账流。流捕获提交到分类账的日记账的每个文档修订，并将数据传送到指定的 Amazon Kinesis Data Streams 资源。

- 默认情况下，Kinesis Data Streams 中的记录聚合处于启用状态。此选项使 QLDB 能够在单个 Kinesis 数据流记录中发布多个数据记录，从而增加每个 API 调用发送的记录数量。

  记录聚合对记录处理具有重要影响，并且需要在流使用者中取消聚合。要了解更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》 中的 KPL 主要概念以及使用者取消聚合。

- `DescribeJournalKinesisStream` — 返回有关给定 QLDB 日记账流的详细信息。输出包括 ARN、流名称、当前状态、创建时间和原始流创建请求的参数。
- `ListJournalKinesisStreamsForLedger`— 返回给定分类账的所有 QLDB 日记账流描述符的列表。每个流描述符的输出都包含返回的`DescribeJournalKinesisStream`相同细节。
- `CancelJournalKinesisStream`— 结束给定的 QLDB 日记账流。在取消流式传输前，当前状态必须为`ACTIVE`。

  停止历程后，您无法重新启动历程。要恢复向 Kinesis Data Streams 传输数据，您可以创建一个新的 QLDB 流。

有关这些 API 操作的完整介绍，请参阅 Amazon QLDB API 参考。

有关使用创建和管理日记流的信息 AWS CLI，请参阅AWS CLI 命令参考。

示例应用程序

QLDB 提供了演示使用日记账流进行各种操作的示例应用程序。这些应用程序在AWS 示例 GitHub 网站上是开源的。

主题

- 基本操作（Java）
- 与 OpenSearch 服务集成 (Python)
- 与 Amazon SNS 和 Amazon SNS 和 Amazon SSS（Python）集成

基本操作（Java）

有关演示 QLDB 日记流基本操作的 Java 代码示例，请参阅 GitHub 存储库 aws-samples/-java。amazon-qldb-dmv-sample有关如何下载和安装此示例应用程序的说明，请参阅 安装 Amazon QLDB Java 示例应用程序。

> **Note**
>
> 安装应用程序后，不要继续执行 Java 教程的步骤 1 来创建分类账。此流媒体示例应用程序将为您创建 `vehicle-registration`分 类账。

此示例应用程序打包了来自 Java 教程 及其依赖项的完整源代码，包括以下模块：

- AWS SDK for Java— 创建和删除 QLDB 和 Kinesis Data Streams 资源，包括分类账、QLDB 日记账流和 Kinesis 数据流。
- 适用于 Java 的 Amazon QLDB 驱动程序— 使用 PartiQL 语句在分类账上运行数据事务，包括创建表和插入文档。
- Kinesis 客户端库：使用和处理 Kinesis 数据流中的数据。

运行代码

该StreamJournal类包含演示以下操作的教程代码：

1. 创建名为的分类账`vehicle-registration`，创建表，然后在其中加载示例数据。

   > **Note**
   >
   > 在运行此代码之前，请确保您还没有名为 `vehicle-registration` 的活动分类账。

2. 创建 Kinesis 数据流、一个允许 QLDB 担任 Kinesis 数据流写入权限的 IAM 角色和 QLDB 日记账流。
3. 使用 KCL 启动流读取器，该读取器处理 Kinesis 数据流并记录每条 QLDB 数据记录。
4. 使用流数据来验证 `vehicle-registration` 示例分类账的哈希链。
5. 通过停止流读取器、取消 QLDB 日记账流、删除分类账和删除 Kinesis 数据流来清理所有资源。

要运行`StreamJournal`教程代码，请从项目根目录中输入以下 Gradle 命令。

```
./gradlew run -Dtutorial=streams.StreamJournal
```

与 OpenSearch 服务集成 (Python)

有关演示如何将 QLDB 流与 OpenSearch 亚马逊服务集成的 Python 示例应用程序，请参阅 GitHub 存储库 aws-samples/-。amazon-qldb-streaming-amazon opensearch-service-sample-python此应用程序使用 AWS Lambda 函数来实现 Kinesis Data Streams 使用者。

在命令行输入以下 `git` 命令以克隆存储库。

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python.git
```

要运行示例应用程序，请参阅上的自述文件 GitHub 以获取说明。

与 Amazon SNS 和 Amazon SNS 和 Amazon SSS（Python）集成

有关演示如何将 QLDB 流与亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 集成的 Python 示例应用程序，请参阅存储库 aws-samples/-。 GitHub amazon-qldb-streams-dmv sample-lambda-python

此应用程序使用 AWS Lambda 函数来实现 Kinesis Data Streams 使用者。它向 Amazon SNS 主题发送消息，该主题已订阅 Amazon Simple Queue Service（Amazon SQS）队列。

在命令行输入以下 `git` 命令以克隆存储库。

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-python.git
```

要运行示例应用程序，请参阅上的自述文件 GitHub 以获取说明。

QLDB 在 Kinesis 中流记录

Amazon QLDB 流向给定的 Amazon Kinesis Data Streams 资源写入三种类型的数据记录：控件、区块摘要和修订详情。所有三种记录类型均以 Amazon Ion 格式的二进制表示形式写入。

控制记录显示您的 QLDB 流的开始和完成。每当有修订提交到您的日记账时，QLDB 流都会写入所有相关的日记账块数据，包括块摘要和修订详细信息记录。

这三种记录类型是多态的。它们都由一条通用的顶级记录组成，其中包含 QLDB 流 ARN、记录类型和记录有效载荷。此顶级记录采用以下格式。

```
{
```

```
  qldbStreamArn: string,
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

recordType 字段可能具有三个值：

- CONTROL
- BLOCK_SUMMARY
- REVISION_DETAILS

以下各节描述每个单独有效载荷记录的格式和内容。

> **Note**
>
> QLDB 以 Amazon Ion 的二进制表示形式将所有流记录写入 Kinesis Data Streams。在 Ion 的文本表示形式中提供了以下示例，以可读格式说明记录内容。

主题

控制层面

QLDB 流写入控制记录以指示其开始和完成事件。以下是每个 controlRecordType 的控制记录示例，包含样本数据：

- CREATED— QLDB 流写入 Kinesis 的第一条记录，表示您新创建的流处于活动状态。

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
```

```
  payload:{
    controlRecordType:"CREATED"
  }
}
```

- COMPLETED— QLDB 流写入 Kinesis 的最后一条记录，表示您的流已到达指定的结束日期和时间。如果您取消流，则不会写入此记录。

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

屏蔽摘要记录

区块摘要记录表示提交您的文档修订的日记账区块。区块是在事务过程中提交到您的 QLDB 日记账的对象。

区块摘要记录的有效载荷包含提交块的块地址、时间戳和其他元数据。它还包括块中修订的摘要属性以及提交这些修订的 PartiQL 语句。以下是包含示例数据的区块摘要记录的示例。

> **Note**
>
> 此区块摘要示例仅供参考。显示的哈希值并非实际计算的哈希值。

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

在 revisionSummaries 字段，某些修订可能没有 documentId。这是仅供内部使用的系统修订版，不包含用户数据。QLDB 流将这些修订包含在各自的区块摘要记录中，因为这些修订的哈希值是日记完整哈希链的一部分。加密验证需要完整的哈希链。

只有具有文档 ID 的修订才会在单独的修订详细信息记录中发布，如下节所述。

修订详细信息记录

修订详细信息记录表示提交到您的日记的文档修订。有效载荷包含修订的提交视图中的所有属性，以及相关的表名称和表ID。以下是带有示例数据的修订记录的示例。

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

处理重复和 out-of-order记录

QLDB 直播可以向 Kinesis Data Streams 发布副本 out-of-order 和记录。因此，使用者应用程序可能需要实现自己的逻辑来识别和处理此类场景。区块摘要和修订详细信息记录包括可用于此目的的字段。结合下游服务的特征，这些字段既可以表示记录的唯一身份，也可以表示严格的记录顺序。

例如，假设一个将 QLDB 与索引集成的流，以提供 OpenSearch 对文档的全文搜索功能。在此用例中，您需要避免索引文档的陈旧 (out-of-order) 修订版。要强制执行排序和重复数据删除，您可以使用中的文档 ID 和外部版本字段 OpenSearch，以及修订详细信息记录中的文档 ID 和版本字段。

有关将 QLDB 与 OpenSearch Amazon Service 集成的示例应用程序中的重复数据删除逻辑示例，GitHub 请参阅存储库 aws-samples/-。amazon-qldb-streaming-amazon opensearch-service-sample-python

QLDB 中的流权限

在创建 Amazon QLDB 流之前，您必须向 QLDB 提供对您指定的 Amazon Kinesis Data Streams 资源的写入权限。如果您正在使用客户托管的 AWS KMS key 来对 Kinesis 流进行服务器端加密，您还必须授予 QLDB 使用您指定的对称加密密钥的权限。Kinesis Data Streams 不支持非对称 KMS 密钥。

要向您的 QLDB 流提供必要的权限，您可以让 QLDB 扮演具有相应权限策略的 IAM 服务角色。服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务委派权限的角色。

> **Note**
>
> 要在请求日记账流时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 `iam:PassRole` 操作的权限。这是对 QLDB 流子资源的 `qldb:StreamJournalToKinesis` 权限的补充。要了解如何使用 IAM 控制对 QLDB 的访问权限，请参阅 Amazon MQ 如何与 IAM 协同工作。有关 QLDB 策略示例，请参阅Amazon QLDB 基于身份的策略示例。

在此示例中，您创建了一个角色，允许 QLDB 代表您向 Kinesis 数据流写入数据记录。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务委派权限的角色。

如果您是首次在中 AWS 账户 流式传输 QLDB 日记，则必须先通过执行以下操作创建具有相应策略的 IAM 角色。或者，您可以使用 QLDB 控制台自动为您创建角色。否则，您可以选择之前创建的角色。

主题

- 创建权限策略
- 创建 IAM 角色

创建权限策略

完成以下步骤，为 QLDB 流创建权限策略。此示例显示了 Kinesis Data Streams 策略，该策略授予 QLDB 向指定的 Kinesis 数据流写入数据记录的权限。如果适用，示例还显示了一个密钥策略，允许 QLDB 使用您指定的对称加密 KMS 密钥。

有关更多信息，请参阅 《Amazon Kinesis Data Streams 开发人员指南》 中的 使用 IAM 控制对 Amazon Kinesis Data Streams 资源的访问 和 授权使用用户生成的 KMS 密钥 要了解有关 AWS KMS 密钥策略的更多信息，请参阅AWS Key Management Service 开发人员指南 AWS KMS中的使用密钥策略。

> **Note**
>
> 您的 Kinesis 数据流和 KMS 密钥必须 AWS 区域 与您的 QLDB 账本位于同一账号中。

使用 JSON 策略编辑器创建策略

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。

2. 在左侧的导航栏中，选择 Policies（策略）。

   如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择开始使用。

3. 在页面的顶部，选择 Create Policy（创建策略）。

4. 请选择 JSON 选项卡。

5. 输入 JSON 策略文档。

   - 如果您正在使用客户托管的 KMS 密钥对 Kinesis 流进行服务器端加密，请使用以下示例策略文档。*###################### us-ea st-1#123456789012 # 1234abcd-12 ab -34cd-56ef-1234567890 kinesis-stream-nameab#*

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

   - 否则，下面是示例策略文档。要使用此政策，请将 us-ea *st-1 #* 123456789012 和示例中的信息替换为您自己的信息。*kinesis-stream-name*

```
{
    "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. 选择Review policy（查看策略）。

> **Note**
>
> 您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy（查看策略），IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在 Review policy（查看策略）页面上，为创建的策略输入 Name（名称）和 Description（说明）（可选）。查看策略摘要以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。

创建 IAM 角色

为您的 QLDB 流创建权限策略后，您可以创建一个 IAM 角色并将您的策略附加到该角色上。

创建用于 QLDB 的服务角色（IAM 控制台）

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，请选择 QLDB，然后选择 QLDB 使用案例。
5. 选择下一步。
6. 选中您之前创建的策略旁边的方框。
7. （可选）设置权限边界。这是一项高级特征，可用于服务角色，但不可用于服务相关角色。

   a. 打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。

      IAM 包含您账户中的 AWS 托管策略和客户托管策略列表。

   b. 选择要用于权限边界的策略。
8. 选择下一步。

9. 输入有助于识别角色的作用的角色名称或者角色名称后缀。

> **Important**
>
> 命名角色时，请注意以下事项：
>
> - 角色名称在您内部必须是唯一的 AWS 账户，并且不能因大小写而变得唯一。
>
> 例如，不要同时创建名为 **PRODROLE** 和 **prodrole** 的角色。当角色名称在策略中使用或者作为 ARN 的一部分时，角色名称区分大小写，但是当角色名称在控制台中向客户显示时（例如，在登录期间），角色名称不区分大小写。
> - 创建角色后，您无法编辑该角色的名称，因为其他实体可能会引用该角色。

10. （可选）对于描述，输入角色的描述。
11. （可选）要编辑角色的使用案例和权限，请在步骤 1：选择可信实体或步骤 2：添加权限部分中选择编辑。
12. （可选）为了帮助识别、组织或搜索角色，请以键值对形式添加标签。有关在 IAM 中使用标签的更多信息，请参阅《IAM 用户指南》中的标记 IAM 资源。
13. 检查该角色，然后选择创建角色。

以下的 JSON 文档是一个信任策略的示例，该策略允许 QLDB 假定一个附有特定权限的 IAM 角色。

```
{
    "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**
>
> 以下示例演示如何使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键来防范混淆代理问题。使用此信任策略，QLDB 只能为分类账 `123456789012` 账户 `myExampleLedger` 中的任何 QLDB 流扮演角色。
> 有关更多信息，请参阅防止跨服务混淆座席。

创建 IAM 角色后，返回 QLDB 控制台并刷新“创建 QLDB 流”页面，以便它可以找到您的新角色。

QLDB 中日记账流的常见错误

本节介绍了 Amazon QLDB 为日记账流请求引发的运行时系统错误。

以下是该服务返回的常见异常列表。每个异常都包括特定的错误消息，然后是简短的描述和可能的解决方案建议。

AccessDeniedException

`######UserArn #####iam#PassRole #####ro Learn`

您无权将 IAM 角色传递给 QLDB 服务。QLDB 要求所有日记账流请求都有一个角色，并且您必须具有将该角色传递给 QLDB 的权限。该角色为 QLDB 提供了在您指定的 Amazon Kinesis Data Streams 资源中的写入权限。

核实您定义了一个 IAM policy，该策略授予对您指定的 QLDB 服务（`qldb.amazonaws.com`）的 IAM 角色资源执行 `PassRole` API 操作的权限。有关策略示例，请参阅Amazon QLDB 基于身份的策略示例。

IllegalArgumentException

消息：QLDB 在验证 Kinesis 数据流时遇到错误：来自 Kinesis 的响应：*`errorCode errorMessage`*

造成此错误的一个可能原因是所提供的 Kinesis 数据流资源不存在。或者，QLDB 没有足够的权限将数据记录写入您指定的 Kinesis 数据流。

验证您在流请求中提供的 Kinesis 数据流是否正确。有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的 创建和更新数据流。

此外，请确认您是否为指定的 Kinesis 数据流定义了一个策略，该策略授予 QLDB 服务（`qldb.amazonaws.com`）对以下操作的权限。有关更多信息，请参阅 流权限。

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

消息：在验证 Kinesis 配置时，来自 Kinesis 数据流的意外响应。来自 Kinesis 的回复：*`errorCode errorMessage`*

尝试将数据记录写入所提供的 Kinesis 数据流失败，并显示提供的 Kinesis 错误响应。有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的 Amazon Kinesis Data Streams 生产人员故障排除。

IllegalArgumentException

消息：开始日期不得早于结束日期。

`InclusiveStartTime` 和 `ExclusiveEndTime` 必须采用 ISO 8601 日期和时间格式以及通用协调时间（UTC）。

IllegalArgumentException

消息：开始日期不能是未来的日期。

`InclusiveStartTime` 和 `ExclusiveEndTime` 都必须采用 `ISO 8601` 日期和时间格式以及 UTC。

LimitExceededException

消息：已超出 Kinesis 数据流同时运行 5 个日记账流的限制

QLDB 强制使用五个并发日记账流的默认限制。

Amazon QLDB 中的分类账管理

本章介绍如何使用 QLDB API、AWS Command Line Interface (AWS CLI)、AWS CloudFormation在 Amazon QLDB 中执行分类账管理操作。

您还可以使用 AWS Management Console 执行这些任务。有关更多信息，请参阅通过控制台访问 Amazon QLDB。

主题

Amazon QLDB 分类账的基本操作

您可以使用 QLDB API 或 AWS Command Line Interface (AWS CLI) 在 Amazon QLDB 中创建、更新和删除分类账。您还可以列出您账户中所有的分类帐，或者获取有关某个分类帐的信息。

以下主题提供了简短的代码示例，这些示例显示了使用AWS SDK for Java和AWS CLI进行分类账操作的常见步骤。

主题

有关在完整示例应用程序中演示这些操作的代码示例，请参阅以下 驱动程序入门 教程和 GitHub 存储库：

- Java：教程 | GitHub 存储库
- Node.js：教程 | GitHub 存储库
- Python：教程 | GitHub 存储库

创建分类账

使用 `CreateLedger` 操作可以在 AWS 账户 中创建分类账。必须提供以下信息：

- 分类帐名称 - 您要在账户中创建的分类帐的名称。该名称在当前 AWS 区域的所有分类帐中必须是唯一的。

  分类账名称的命名约束在Amazon QLDB 资源中的限额和限制中定义。
- 权限模式 - 要分配给分类帐的权限模式。请选择以下任一选项：
  - 允许所有 – 这是一种旧式权限模式，支持对分类账进行 API 级别粒度的访问控制。

    此模式允许拥有此分类账的 `SendCommand` API 权限的用户在指定分类账中的任何表上运行所有 PartiQL 命令(因此，`ALLOW_ALL`)。此模式忽略您为分类账创建的任何表级或命令级 IAM 权限策略。
  - 标准 – (推荐) 一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。我们强烈建议使用此权限模式来最大限度地提高分类账数据的安全性。

    默认情况下，此模式拒绝所有用户请求在此分类账中的任何表上运行任何 PartiQL 命令。要允许 PartiQL 命令运行，除了分类账的 `SendCommand` API 权限外，您还必须为特定表资源和 PartiQL 操作创建 IAM 权限策略。有关信息，请参阅 请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。
- 删除保护 – (可选) 防止分类账被任何用户删除的标志。如果创建分类账期间未定义，则默认情况下启用该功能(`true`)。

  如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 `UpdateLedger` 操作将标志设置为 `false`，来禁用它。
- AWS KMS key—(可选)在 AWS Key Management Service(AWS KMS) 中用于加密静态数据的密钥。选择以下AWS KMS keys类型之一：
  - AWS 拥有的 KMS 密钥 – 使用 AWS 代表您拥有和管理的 KMS 密钥。

    如果在分类帐创建过程中未定义此参数，则分类帐默认使用此类型的密钥。您还可以使用字符串 `AWS_OWNED_KMS_KEY` 指定此密钥类型。此选项不要求其他设置。
  - A valid symmetric customer managed KMS key(有效的对称客户托管 KMS 密钥)：在您创建、拥有和管理的账户中使用指定的对称加密 KMS 密钥。QLDB 不支持非对称密钥。

    此选项要求您创建 KMS 密钥或使用账户中现有密钥。有关创建客户托管密钥的说明，请参阅 AWS Key Management Service《开发者指南》 中的 创建对称加密 KMS 密钥。

要指定客户托管的 KMS 密钥，您可以使用其密钥 ID、别名或 Amazon 资源名称(ARN)。有关更多信息，请参阅 AWS Key Management Service《开发者指南》 中的 密钥标识符(KeyId)。

> **Note**
>
> 不支持跨区域密钥。指定的 KMS 密钥必须与您的分类账处于同一 AWS 区域 中。

有关更多信息，请参阅Amazon QLDB 中的静态加密。

- 标签 – (可选) 通过以键值对的形式附加标签来向分类账添加元数据。您可以向分类账添加标签来帮助组织和标识它们。有关更多信息，请参阅为 Amazon QLDB 资源贴标签。

在 QLDB 创建表并且将其状态设置 `ACTIVE` 之前，分类账将无法使用。

创建分类帐 (Java)

要使用 AWS SDK for Java 创建分类账

1. 创建 `AmazonQLDB` 类的实例。
2. 创建 `CreateLedgerRequest` 类实例，以提供请求信息。

   您必须提供分类账名称和权限模式。
3. 以参数形式提供请求对象，运行 `createLedger` 方法。

该 `createLedger` 请求返回包含分类账信息的 `CreateLedgerResult` 对象。有关在创建分类账后使用 `DescribeLedger` 操作检查分类账状态的示例，请参阅下一节。

以下示例演示了上述步骤。

Example — 使用默认配置设置

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> 如果您未指定默认设置，分类账将使用以下默认设置：
>
> - 删除保护 – 激活 (`true`)。
> - KMS 密钥 — AWS 拥有的 KMS 密钥。

Example — 禁用删除保护，使用客户托管 KMS 密钥并附加标签

```
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);
```

创建分类账 (AWS CLI)

使用默认配置设置创建名为`vehicle-registration`的新分类账。

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> 如果您未指定默认设置，分类账将使用以下默认设置：
>
> - 删除保护 – 激活 (`true`)。
> - KMS 密钥 — AWS 拥有的 KMS 密钥。

或者使用指定的客户托管 KMS 密钥和指定标签创建名为vehicle-registration的新分类账，该新分类账禁用删除保护。

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
```

创建分类账 (AWS CloudFormation)

您也可以使用 AWS CloudFormation 模板创建分类账。有关更多信息，请参阅 AWS::QLDB::Ledger resource in the AWS CloudFormation User Guide。

描述分类帐

要查看有关分类账的详细信息，请使用 DescribeLedger 操作。您必须提供分类账名称。DescribeLedger 的输出格式与 CreateLedger 相同。其中包含以下信息：

- 分类帐名称 - 您要描述的分类帐的名称。
- ARN – 分类账的 Amazon 资源名称 (ARN)(采用以下格式)。

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

- 删除保护 - 指示是否启用删除保护功能的标志。
- 创建日期和时间 - 创建分类帐时的日期和时间 (采用纪元时间格式)。
- 状态 - 分类账的当前状态。它可以是以下值之一：
  - CREATING
  - ACTIVE
  - DELETING
  - DELETED
- 权限模式 - 要分配给分类帐的权限模式。它可以是以下值之一：
  - ALLOW_ALL：一种旧式权限模式，支持对分类账进行 API 级别粒度的访问控制。

  - STANDARD：一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。
- 加密描述 - 有关分类账中静态数据加密的信息。这些信息包含以下各项：
  - AWS KMS keyARN — 分类账用于静态加密的、客户托管 KMS 的 ARN。如果该密钥是未定义的，分类帐使用 AWS 拥有的 KMS 密钥进行加密。
  - 加密状态 - 分类账静态加密的当前状态。它可以是以下值之一：
    - ENABLED — 使用指定的密钥完全启用加密。
    - UPDATING — 正在积极处理指定的密钥更改。

      QLDB 中的关键更改为异步类型。处理关键变更期间，分类账完全可以访问，不会对性能产生任何影响。秘钥更新所用时间因分类账大小而异。
    - KMS_KEY_INACCESSIBLE — 无法访问指定的客户托管的 KMS 密钥，并且分类账受损。密钥被禁用或删除，或对密钥的授权已被撤销。当分类账受损时，它将无法访问，并且不接受任何读取或写入请求。

      在您恢复对密钥的授权或重新启用已禁用的密钥之后，受损分类账会自动返回至活动状态。但是，删除客户托管的 KMS 密钥操作不可逆。删除密钥后，您无法再访问使用该密钥保护的分类帐，并且该数据将无法永久恢复。
  - 不可访问 AWS KMS key — 出错时 KMS 密钥首次无法访问的日期和时间，采用纪元时间格式。

    如果 KMS 秘钥可访问，则这是未定义。

有关更多信息，请参阅Amazon QLDB 中的静态加密。

> **Note**
>
> 创建 QLDB 分类账后，当其状态从CREATING变为ACTIVE时，该分类账即可使用。

描述分类账 (Java)

使用 AWS SDK for Java 描述分类账

1. 创建 AmazonQLDB 类的实例。或者，您可以使用为AmazonQLDB请求实例化的相同CreateLedger客户端实例。
2. 创建一个 DescribeLedgerRequest 类的实例，并提供您想要描述的分类账的名称。
3. 以参数形式提供请求对象，运行 describeLedger 方法。

4. 该 `describeLedger` 请求返回包含分类账信息的 `DescribeLedgerResult` 对象。

以下代码示例演示了上述步骤。您可以随时调用客户端的 `describeLedger` 方法来获取分类账信息。

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());
```

描述分类账 (AWS CLI)

描述您创建的 `vehicle-registration` 分类帐。

Example

```
aws qldb describe-ledger --name vehicle-registration
```

更新分类账

该 `UpdateLedger` 操作目前允许您更改现有分类账的以下配置设置：

- 删除保护 - 防止分类账被任何用户删除的标志。如果启用了此功能，您必须先通过将标记设置为`false`禁用，然后再删除分类账。

  如果未定义此参数，则不需要对分类帐删除保护设置进行任何更改。
- AWS KMS key—（可选）在 AWS Key Management Service(AWS KMS) 中用于加密静态数据的密钥。如果未定义此参数，则不需要对分类帐 KMS 秘钥进行任何更改。

> **Note**
>
> Amazon QLDB 于 2021 年 7 月 22 日推出了对客户托管AWS KMS keys的支持。默认情况下，在发布之前创建的所有分类账都受到AWS 拥有的密钥保护，但目前不符合使用客户托管密钥进行静态加密的资格。
> 您可在 QLDB 控制台查看分类账的创建时间。

使用以下选项之一：

- AWS 拥有的 KMS 密钥 – 使用 AWS 代表您拥有和管理的 KMS 密钥。若要使用这种类型的密钥，请为此参数指定`AWS_OWNED_KMS_KEY`字符串。此选项不要求其他设置。
- A valid symmetric customer managed KMS key(有效的对称客户托管 KMS 密钥)：在您创建、拥有和管理的账户中使用指定的对称加密 KMS 密钥。QLDB 不支持非对称密钥。

  此选项要求您创建 KMS 密钥或使用账户中现有密钥。有关创建客户托管密钥的说明，请参阅 AWS Key Management Service《开发者指南》 中的 创建对称加密 KMS 密钥。

  要指定客户托管的 KMS 密钥，您可以使用其密钥 ID、别名或 Amazon 资源名称(ARN)。有关更多信息，请参阅 AWS Key Management Service《开发者指南》 中的 密钥标识符(KeyId)。

  > **Note**
  >
  > 不支持跨区域密钥。指定的 KMS 密钥必须与您的分类账处于同一 AWS 区域 中。

QLDB 中的关键更改为异步类型。处理关键变更期间，分类账完全可以访问，不会对性能产生任何影响。

您可根据需要随时切换密钥，但是更新密钥所需的时间因分类账大小而异。您可使用 `DescribeLedger` 操作检查静态加密状态。

有关更多信息，请参阅Amazon QLDB 中的静态加密。

`UpdateLedger` 的输出格式与 `CreateLedger` 相同。

更新分类账 (Java)

要使用 AWS SDK for Java 更新分类账

1. 创建 AmazonQLDB 类的实例。
2. 创建 UpdateLedgerRequest 类实例，以提供请求信息。

   您必须提供分类账名称、用于删除保护的新布尔值、或 KMS 密钥新字符串值。
3. 以参数形式提供请求对象，运行 updateLedger 方法。

以下代码示例演示了上述步骤。该 updateLedger 请求返回包含分类账信息的 UpdateLedgerResult 对象。

Example - 禁用删除保护

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example - 使用客户自主管理型密钥

```
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 - 使用 AWS 拥有的 KMS 密钥

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

更新分类账 (AWS CLI)

如果vehicle-registration 分类账检测到启用了删除保护，您必须先禁用它，然后才能删除。

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

您也可以将分类账静态加密设置更改为使用客户托管 KMS 密钥。

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

或者，您可使用AWS拥有的 KMS 秘钥更改静态加密设置。

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

更新分类账权限模式

该 UpdateLedgerPermissionsMode 操作允许您更改现有分类账的权限模式。请选择以下任一选项：

- 允许所有 – 这是一种旧式权限模式，支持对分类账进行 API 级别粒度的访问控制。

  此模式允许拥有此分类账的 SendCommand API 权限的用户在指定分类账中的任何表上运行所有 PartiQL 命令(因此，ALLOW_ALL)。此模式忽略您为分类账创建的任何表级或命令级 IAM 权限策略。

- 标准 – (推荐) 一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。我们强烈建议使用此权限模式来最大限度地提高分类账数据的安全性。

  默认情况下，此模式拒绝所有用户请求在此分类账中的任何表上运行任何 PartiQL 命令。要允许 PartiQL 命令运行，除了分类账的 SendCommand API 权限外，您还必须为特定表资源和 PartiQL 操作创建 IAM 权限策略。有关信息，请参阅 请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

> **Important**
>
> 在切换至 STANDARD 权限模式前，必须先创建所有必需的 IAM 策略和表标签，以避免对用户造成干扰。要了解更多信息，请继续 迁移至标准权限模式。

更新分类账权限模式 (Java)

要更新分类账权限模式，请使用 AWS SDK for Java

1. 创建 AmazonQLDB 类的实例。
2. 创建 UpdateLedgerPermissionsModeRequest 类实例，以提供请求信息。

   您必须按此权限模式提供分类账名称和新字符串值。
3. 以参数形式提供请求对象，运行 updateLedgerPermissionsMode 方法。

以下代码示例演示了上述步骤。该 updateLedgerPermissionsMode 请求返回包含分类账更新信息的 UpdateLedgerPermissionsModeResult 对象。

Example — 分配标准权限模式

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

更新分类账权限模式 (AWS CLI)

为您的 vehicle-registration 分类账分配 STANDARD 权限模式。

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

迁移至标准权限模式

要迁移至 STANDARD 权限模式，我们建议您分析您的 QLDB 访问模式，并添加 IAM 策略，向您的用户授予访问其资源的相应权限。

在切换至 STANDARD 权限模式前，必须先创建所有必需的 IAM 策略和表标签。否则，在您创建正确的 IAM 策略或将权限模式恢复到ALLOW_ALL之前，切换权限模式可能会干扰用户。有关创建这些策略的更多信息，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

您也可以使用 AWS 策略授予所有 QLDB 资源的完全访问权限。AmazonQLDBFullAccess 和 AmazonQLDBConsoleFullAccess 托管策略包含所有 QLDB 操作，包括所有 PartiQL 操作。将其中一个策略附加至主体，等同于该委托人的 ALLOW_ALL 权限模式。有关更多信息，请参阅AWS 亚马逊 QLDB 的托管策略。

删除分类帐

使用 DeleteLedger 操作删除分类帐及其所有内容。分类账的删除操作是不可恢复的。

如果分类账启用了删除保护，您必须先禁用它，然后才能删除分类账。

在您发出 DeleteLedger 请求时，分类账的状态从 ACTIVE 变为 DELETING。删除分类账可能需要一段时间，具体取决于分类账使用的存储量。在 DeleteLedger 操作结束时，该分类账在 QLDB 中不再存在。

删除分类账 (Java)

要使用 AWS SDK for Java 级的分类账

1. 创建 AmazonQLDB 类的实例。
2. 创建一个 DeleteLedgerRequest 分类账的实例，并提供您想要删除的分类账的名称。
3. 以参数形式提供请求对象，运行 deleteLedger 方法。

以下代码示例演示了上述步骤。

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

删除分类账 (AWS CLI)

删除您的vehicle-registration分类账。

Example

```
aws qldb delete-ledger --name vehicle-registration
```

列出分类账

该 ListLedgers 操作返回当前AWS 账户和区域的所有 QLDB 分类账摘要信息。

列出分类账 (Java)

要列出您账户中的分类账，请使用 AWS SDK for Java

1. 创建 AmazonQLDB 类的实例。
2. 创建 ListLedgersRequest 类的实例。

   如果您在上一次 ListLedgers 调用的响应中收到了NextToken值，则必须在此请求中提供该值才能获得下一页结果。
3. 以参数形式提供请求对象，运行 listLedgers 方法。
4. 该 listLedgers 请求返回ListLedgersResult对象。此对象包含LedgerSummary对象列表和分页令牌，用于指示是否还有更多结果可用：

   - 如果 NextToken 为空，则表示最后一页结果已处理完毕，没有其他结果。
   - 如果NextToken不为空，则表示有更多结果可用。若要检索下一页结果，请在后续 ListLedgers 调用中使用NextToken值。

以下代码示例演示了上述步骤。

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);
```

列出分类账 (AWS CLI)

列出当前 AWS 账户 和区域中的所有分类账。

Example

```
aws qldb list-ledgers
```

使用 AWS CloudFormation 创建 Amazon QLDB 资源

Amazon QLDB 与 AWS CloudFormation 集成，后者是一项服务，可帮助您对 AWS 资源进行建模和设置，这样您只需花较少的时间来创建和管理资源与基础设施。您可以创建一个描述所需的全部 AWS 资源的模板(例如您的 QLDB 分类账)，AWS CloudFormation 为您预置和配置这些资源。

在您使用 AWS CloudFormation 时，可重复使用您的模板来不断地重复设置您的 QLDB 资源。描述您的资源一次，然后在多个 AWS 账户 和区域中反复预置相同的资源。

QLDB 和AWS CloudFormation模板

要为 QLDB 和相关服务设置和配置资源，您必须了解 AWS CloudFormation模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 AWS CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 AWS CloudFormation Designer 的帮助下开始使用 AWS CloudFormation 模板。有关更多信息，请参阅 AWS CloudFormation 用户指南中的什么是 AWS CloudFormation Designer？。

QLDB 支持在AWS CloudFormation中创建分类账和日志账流。有关更多信息，包括分类账和流的 JSON 和 YAML 模板示例，请参阅 AWS CloudFormation用户指南 中的下述资源类型参考：

- AWS:: QLDB:: 分类账参考
- AWS:: QLDB:: 流引用

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation API 参考
- AWS CloudFormation 命令行界面用户指南

为 Amazon QLDB 资源贴标签

标签是您分配的自定义属性标签，或者是 AWS 分配给 AWS 资源的标签。每个标签具有两个部分：

- 标签键（例如，`CostCenter`、`Environment` 或 `Project`）。标签键区分大小写。
- 一个称为标签值 的可选字段（例如，`111122223333` 或 `Production`）。省略标签值与使用空字符串效果相同。与标签键一样，标签值区分大小写。

标签可帮助您：

- 标识和整理您的 AWS 资源。许多 AWS 服务都支持添加标签，因此，您可以将同一标签分配给来自不同服务的资源，以表明这些资源相互之间存在关系。例如，您可以将相同的标签分配给为 Amazon S3 存储桶分配的 QLDB 分类账。
- 跟踪您的 AWS 成本。您可以在 AWS Billing and Cost Management 控制面板上激活这些标签。AWS 使用标签对您的成本进行分类，并向您提供每月成本分配报告。有关更多信息，请参阅 AWS Billing 用户指南中的使用成本分配标签。
- 使用 AWS Identity and Access Management (IAM) 控制对 AWS 资源的访问。有关信息，请参阅QLDB 基于属性的访问控制（ABAC）本开发者指南和 IAM 用户指南中的使用 IAM 标签控制访问权限。

有关使用标签的提示，请参阅AWS回答博客上的AWS标记策略文章。

以下各部分提供有关 Amazon QLDB 标签的更多信息。

主题

Amazon QLDB 中支持的资源

Amazon QLDB 中的以下资源支持贴标签：

- 分类账
- table

- 日记账流

有关添加和管理标签的信息，请参阅管理标签。

标签命名和使用约定

以下基本命名和使用约定适用于将标签与 Amazon QLDB 资源一起使用的情况：

- 每个资源最多可以有 50 个标签。
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大标签键长度为 128 个 Unicode 字符 (采用 UTF-8 格式)。
- 最大标签值长度为 256 个 Unicode 字符 (采用 UTF-8 格式)。
- 允许使用的字符包括可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：. : + = @ _ / - (连字符)。
- 标签键和值区分大小写。最佳实践是，决定利用标签的策略并在所有资源类型中一致地实施该策略。例如，决定是使用 `Costcenter`、`costcenter` 还是 `CostCenter`，以及是否对所有标签使用相同的约定。避免将类似的标签用于不一致的案例处理。
- `aws:` 前缀专门预留供 AWS 使用。当标签具有带 `aws:` 前缀的标签键时，您将无法编辑或删除标签的键或值。具有此前缀的标签不计入每个资源的标签数限制。

管理标签

标签由资源上的 `Key` 和 `Value` 属性构成。可以使用 Amazon QLDB 控制台、AWS CLI 或 QLDB API 添加、编辑或删除这些属性的值。您还可以使用 AWS Resource Groups 标签编辑器 来管理标签。

有关使用标记的信息，请参阅以下 API 操作：

- Amazon QLDB API 参考中的 ListTagsForResource
- Amazon QLDB API 参考中的 TagResource
- Amazon QLDB API 参考中的 UntagResource

使用 QLDB 标记面板（控制台）

1. 登录 AWS Management Console 并打开 Amazon QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。
2. 在导航窗格中，选择分类账。

3. 在分类账列表中，选择要管理其标签的分类账名称。
4. 在分类账详细信息页面上，找到标记卡片并选择管理标记。
5. 在管理标签页面上，您可以根据自己的分类账添加、编辑或删除任何标记。当标记键和值符合您的要求时，选择保存。

在创建时标记资源

对于支持标记的 QLDB 资源，您可以在创建资源时使用AWS CLI、或 QLDB AP AWS Management Console I 定义标签。通过在创建资源时对其进行标记，无需在创建资源后运行自定义标记脚本。

对资源进行标记后，您可根据这些标签来控制对资源的访问。例如：您只能向具有特定标签的表资源授予完全访问权限。有关 JSON 策略示例，请参阅基于表格标签对所有操作的完全访问权限。

> **Note**
>
> 表和流资源不会继承其根分类账资源的标签。

还可以通过在 `CREATE TABLE` PartiQL 语句中指定表标记来定义表标记。要了解更多信息，请参阅“为表格添加标签”。

Amazon QLDB 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方 AWS 的共同责任。责任共担模式将其描述为云的 安全性和云中 的安全性：

- 云安全 — AWS 负责保护在云 AWS 服务 中运行的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 Amazon ECR 的合规性计划，请参阅合规性计划范围内的AWS 服务。
- 云端安全 — 您的责任由您 AWS 服务 使用的内容决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 QLDB 时应用责任共担模式。以下主题说明如何配置 QLDB 以实现您的安全性和合规性目标。您还将学习如何使用其他方法 AWS 服务 来帮助您监控和保护您的 QLDB 资源。

主题

Amazon QLDB 中的数据保护

责任 AWS 共担模式分适用于 Amazon QLDB 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API AWS CLI或 SDK 使用 QLDB AWS 服务 或其他软件开发工具包的情况。 AWS 在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

> **Note**
>
> 关于避免在敏感信息中使用标签或自由格式字段的建议是指 QLDB 分类账资源的元数据，而不是存储在账本内的数据。您在 QLDB 分类账资源内存储的数据不用于计费或诊断日记账。

主题

Amazon QLDB 中的静态加密

Amazon DynamoDB 中存储的所有用户数据在静态状态下进行完全加密。QLDB 静态加密通过使用 () 中的加密密钥对所有静态账本数据进行加密，从而增强安全性。 AWS Key Management Service AWS KMS此功能减少保护敏感数据时涉及的操作负担和复杂性。利用静态加密，可以构建符合严格加密合规性和法规要求的安全敏感型应用程序。

静态加密与集成， AWS KMS 用于管理用于保护您的 QLDB 账本的加密密钥。有关的更多信息 AWS KMS，请参阅《AWS Key Management Service 开发人员指南》中的AWS Key Management Service 概念。

在 QLDB 中，您可以为每个账本资源指定类型 AWS KMS key 。创建新分类账或更新现有分类账时，您可以选择以下类型的 KMS 密钥之一来保护您的分类账数据：

- AWS 拥有的密钥 – 默认加密类型。此密钥归 QLDB 拥有（不另外收费）。
- 客户托管的密钥 - 此密钥存储在您的 AWS 账户 中，由您创建、拥有和托管。您可以完全控制钥匙（AWS KMS 收费）。

> **Note**
>
> 亚马逊 QLDB 于 2021 年 7 月 22 日推出了对客户 AWS KMS keys 管理的支持。默认情况下，在发布之前创建的所有账本都 AWS 拥有的密钥 受到保护，但目前不符合使用客户托管密钥进行静态加密的资格。
> 您可在 QLDB 控制台查看分类账的创建时间。

当您访问分类账时，QLDB 会以透明方式解密表数据。您可以随时在客户管理的密钥 AWS 拥有的密钥和客户管理的密钥之间切换。无需更改任何代码或应用程序即可使用或管理加密表。

您可以在创建新账本时指定加密密钥，也可以使用 QLDB API 或 () 更改现有账本的加密密钥。 AWS Management Console AWS Command Line Interface AWS CLI有关更多信息，请参阅 在 Amazon QLDB 中使用客户托管的密钥。

> **Note**
>
> 默认情况下，Amazon QLDB 会自动使用 AWS 拥有的密钥 免费启用加密。但是，使用客户管理的密钥需要 AWS KMS 付费。有关定价的信息，请参阅 AWS Key Management Service 定价。
> QLDB 静态加密可在所有可用 QLDB AWS 区域 的地方使用。

主题

- Amazon QLDB 静态加密：工作原理
- 在 Amazon QLDB 中使用客户托管的密钥

Amazon QLDB 静态加密：工作原理

QLDB 静态加密 静态加密使用 256 位高级加密标准（AES-256）来加密数据。这有助于保护您的数据，防止对底层存储进行未经授权的访问。默认情况下，存储在 QLDB 分类账中的所有数据都是静态加密。服务器端加密是透明的，这意味着不需要对应用程序进行更改。

静态加密与 AWS Key Management Service (AWS KMS) 集成，用于管理用于保护 QLDB 账本的加密密钥。在创建新账本或更新现有账本时，您可以选择以下类型的 AWS KMS ：

- AWS 拥有的密钥 – 默认加密类型。此密钥归 QLDB 拥有（不另外收费）。
- 客户托管的密钥 - 此密钥存储在您的 AWS 账户 中，由您创建、拥有和托管。您可以完全控制钥匙（AWS KMS 收费）。

主题

AWS 拥有的密钥

AWS 拥有的密钥 没有存储在你的 AWS 账户。它们是 KMS 密钥集合的一部分，这些密钥 AWS 拥有并管理多个密钥 AWS 账户。 AWS 服务 可以 AWS 拥有的密钥 用来保护您的数据。

您无需创建或管理 AWS 拥有的密钥。但是，您无法查看 AWS 拥有的密钥、跟踪或审核它们的使用情况。您无需支付月费或使用费 AWS 拥有的密钥，也不会计入您账户的 AWS KMS 配额。

有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS 拥有的密钥。

客户托管密钥

客户托管密钥是您 AWS 账户 自己创建、拥有和管理的 KMS 密钥。您对 KMS 密钥拥有全部控制权。仅支持对称加密 KMS 密钥。

使用客户托管密钥可获得以下功能：

- 使用密钥策略、IAM policy 和授权以控制对 KMS 密钥的访问

- 启用和禁用密钥
- 轮换密钥的加密材料
- 创建密钥标签和别名
- 安排删除 KMS 密钥
- 导入您自己的密钥材料或使用您拥有和托管的自定义密钥存储
- 使用 AWS CloudTrail 和 Amaz CloudWatch on Logs 来跟踪 QLDB 代表您发送 AWS KMS 的请求

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户托管密钥。

客户托管的密钥每次 API 调用都会产生费用，而 AWS KMS 配额适用于这些 KMS 密钥。有关更多信息，请参阅AWS KMS 资源或请求限额。

当您将一个客户管理的密钥指定为账本的 KMS 密钥时，日记账存储和索引存储中的所有账本数据都会受到相同的客户管理的密钥保护。

无法访问客户托管的密钥

如果您禁用客户托管密钥、计划删除密钥或撤销对密钥的授权，则分类账加密状态将变为 `KMS_KEY_INACCESSIBLE`。在这种状态下，分类账会受损，不接受任何读取或写入请求。无法访问的密钥会阻止所有用户和 QLDB 服务加密或解密数据，也无法在分类账中执行读写操作。QLDB 必须有权访问加密密钥，确保您可以继续访问表并防止数据丢失。

> **Important**
>
> 在您恢复对密钥的授权或重新启用已禁用的密钥之后，受损分类账会自动返回至活动状态。但是，删除客户托管的密钥是不可逆的。删除密钥后，您将无法再访问使用该密钥保护的分类账，数据将永久无法恢复。

要检查账本的加密状态，请使用 AWS Management Console 或 DescribeLedgerAPI 操作。

Amazon QLDB 如何使用授权 AWS KMS

QLDB 需要授权，才能使用客户托管密钥。当您创建受客户托管密钥保护的账本时，QLDB 会通过向发送请求来CreateGrant代表您创建授权。 AWS KMS中的授权 AWS KMS 用于让 QLDB 访问客户中的 KMS 密钥。 AWS 账户有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用授权。

QLDB 无服务器需要授权，才能将客户托管的密钥用于以下 AWS KMS 操作：

- DescribeKey— 验证指定的对称加密 KMS 密钥是否有效。
- GenerateDataKey— 生成唯一的对称数据密钥，QLDB 使用该密钥对账本中的静态数据进行加密。
- Decrypt - 解密客户托管式密钥加密的数据。
- 加密 - 使用您的客户托管密钥将明文加密为密文。

您可以随时撤消针对授权的访问权限，也可以移除服务访问客户托管密钥的权限。如果这样做，密钥将无法访问，QLDB 将无法访问受客户托管密钥保护的任何分类账数据。在这种状态下，分类账受损，在您恢复对密钥的授权之前，分类账不接受任何读取或写入请求。

恢复 AWS KMS中的授权

要恢复对客户托管密钥的授权并恢复对 QLDB 中分类账的访问权限，您可以对分类账进行更新并指定相同的 KMS 密钥。有关说明，请参阅 更新现有分类账的 AWS KMS key。

使用静态加密时的注意事项

在 QLDB 中使用静态加密时，请注意以下事项：

- 所有 QLDB 分类账数据上都启用了服务器端静态加密，无法禁用。无法仅加密分类账中的子集。
- 静态加密仅加密持久存储介质上的静态数据。如果正在传输的数据或正在使用的数据担心数据安全，则可能需要采取额外措施：
  - 传输中的数据： 中的所有数据都在传输中加密。默认情况下，与 QLDB 的通信将使用 HTTPS 协议，通过安全套接字层（SSL）/传输层安全性（TLS）加密保护网络流量。
  - 使用中的数据：正在使用的数据：在将数据发送到 QLDB 之前，使用客户端加密来保护您的数据。

要了解如何为分类账实现客户托管密钥，请继续 在 Amazon QLDB 中使用客户托管的密钥。

在 Amazon QLDB 中使用客户托管的密钥

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 QLDB API 为 Amazon QLDB 中的新账 AWS KMS key 本和现有账本指定。以下主题介绍如何在 QLDB 中管理和监控客户托管密钥的使用情况。

主题

- 先决条件
- 为新分类账指定 AWS KMS key
- 更新现有分类账的 AWS KMS key
- 监控 AWS KMS keys

先决条件

必须先在 () 中创建密钥，然后才能使用客户托管密钥保护 QLDB 账本。 AWS Key Management Service AWS KMS您还必须指定密钥策略，允许 QLDB 代表您创建授权。 AWS KMS key

在 中创建客户托管的密钥

要创建客户托管密钥，请按照《AWS Key Management Service 开发人员指南》的创建对称加密 KMS 密钥中的步骤进行操作。QLDB 不支持非对称密钥。

设置密钥策略

密钥策略是控制中客户托管密钥访问权限的主要方法 AWS KMS。每个 KMS 密钥必须只有一个密钥策略。密钥策略文档中的语句确定谁有权使用 KMS 密钥以及如何使用它。有关更多信息，请参阅中的使用密钥策略 AWS KMS。

创建客户托管式密钥时，可以指定密钥策略。要更改现有客户托管密钥的密钥策略，请参阅更改密钥策略。

要允许 QLDB 使用您的客户托管密钥，密钥策略必须包含以下操作的权限： AWS KMS

- kms: CreateGrant — 向客户托管密钥添加授权。授予对指定 KMS 密钥的控制访问权限。

  当您使用指定的客户管理密钥创建或更新账本时，QLDB 会创建授权，以允许访问其所需的授权操作。授权操作包括：

  - GenerateDataKey
  - Decrypt
  - Encrypt
- kms: DescribeKey — 返回有关客户托管密钥的详细信息。QLDB 使用此信息来验证密钥。

密钥策略示例

以下是您可以 用于 QLDB 的一个示例密钥策略。此策略略允许经授权使用 QLDB 的账户111122223333委托人调用资源 `arn:aws:kms:us-`

east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab 上的 DescribeKey 和 CreateGrant 操作。

############ us-east-1#111122223333 # 1234abcd-12ab-34cd-56ef-1234567890ab 替换为您自己的信息。

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

为新分类账指定 AWS KMS key

使用 QLDB 控制台或 AWS CLI创建新分类账时，请按照以下步骤指定 KMS 密钥。

要指定客户托管的密钥，您可以使用其密钥 ID、Amazon 资源名称（ARN）、别名或别名 ARN。要了解更多信息，请参阅《AWS Key Management Service 开发者指南》中的密钥标识符 (KeyId)。

> **Note**
>
> 不支持跨区域密钥。指定的 KMS 密钥必须与您的账本 AWS 区域 相同。

创建分类账（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 选择 创建分类账。
3. 在 创建分类账 页面上，执行以下操作：
   - 分类帐信息-输入在当前 AWS 账户 和区域的所有分类帐中唯一的分类帐名称。
   - 权限模式 - 选择要分配给分类帐的权限模式：
     - 允许全部
     - 标准（已推荐）
   - 加密静态数据：选择用于静态加密的 KMS 密钥类型：
     - 使用 AWS 拥有的 KMS 密钥-使用由 AWS 您代表拥有和管理的 KMS 密钥。这是默认选项，无需额外设置。
     - 选择其他 AWS KMS 密钥-在您的账户中使用由您创建、拥有和管理的对称加密 KMS 密钥。

       要使用 AWS KMS 控制台创建新密钥，请选择创建 AWS KMS 密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南 中的创建对称加密 KMS 密钥。

       要使用现有 KMS 密钥，请从下拉列表中选择一个密钥或指定 KMS 密钥 ARN。
4. 根据需要进行设置后，选择 创建分类账。

   当 QLDB 分类账的状态变为“活跃”时，您可以访问该 QLDB 分类账。这个过程可能需要几分钟。

创建分类账（AWS CLI）

使用在 AWS CLI QLDB 中使用 AWS 拥有的密钥 默认密钥或客户托管密钥创建账本。

Example — 使用默认 AWS 拥有的密钥创建分类账

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

Example — 创建包含客户托管密钥的分类账

```
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
```

更新现有分类账的 AWS KMS key

您也可以随时使用 QLDB 控制台或 AWS CLI 将现有账本的 KMS 密钥更新 AWS 拥有的密钥 为或客户托管密钥。

> **Note**
>
> 亚马逊 QLDB 于 2021 年 7 月 22 日推出了对客户 AWS KMS keys 管理的支持。默认情况下，在发布之前创建的所有账本都 AWS 拥有的密钥 受到保护，但目前不符合使用客户托管密钥进行静态加密的资格。
> 您可在 QLDB 控制台查看分类账的创建时间。

QLDB 中的关键更改为异步类型。在处理关键变更期间，分类账完全可以访问，不会对性能产生任何影响。秘钥更新所用时间因分类账大小而异。

要指定客户托管的密钥，您可以使用其密钥 ID、Amazon 资源名称（ARN）、别名或别名 ARN。要了解更多信息，请参阅《AWS Key Management Service 开发者指南》中的密钥标识符 (KeyId)。

> **Note**
>
> 不支持跨区域密钥。指定的 KMS 密钥必须与您的账本 AWS 区域 相同。

创建分类账（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 在导航窗格中，选择分类账。
3. 在分类账列表中，选择要更新的分类帐，然后选择编辑分类账。
4. 在编辑分类账页面上，选择用于静态加密的 KMS 密钥类型：
   - 使用 AWS 拥有的 KMS 密钥-使用由 AWS 您代表拥有和管理的 KMS 密钥。这是默认选项，无需额外设置。
   - 选择其他 AWS KMS 密钥-在您的账户中使用由您创建、拥有和管理的对称加密 KMS 密钥。

要使用 AWS KMS 控制台创建新密钥，请选择创建 AWS KMS 密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南 中的创建对称加密 KMS 密钥。

要使用现有 KMS 密钥，请从下拉列表中选择一个密钥或指定 KMS 密钥 ARN。

5. 选择“确认更改”。

更新分类账（AWS CLI）

使用使用 AWS 拥有的密钥 默认密钥或客户托管密钥更新 QLDB 中的现有账本。 AWS CLI

Example — 使用默认 AWS 拥有的密钥创建分类账

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — 更新包含客户托管密钥的分类账

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

监控 AWS KMS keys

如果您使用客户托管密钥来保护您的 Amazon QLDB 账本，则可以使用AWS CloudTrail或 A CloudWatch mazon Logs 来跟踪 QLDB 代表您发送的请求。 AWS KMS 有关更多信息，请参阅 AWS Key Management Service 开发人员指南 中的 监控 AWS KMS keys。

以下示例是操作CreateGrant、、GenerateDataKeyDecryptEncrypt、和的 CloudTrail 日志条目DescribeKey。

CreateGrant

当您指定客户托管密钥来保护您的账本时，QLDB CreateGrant 会代表您向发送请求 AWS KMS 以允许访问您的 KMS 密钥。此外，当您删除分类账时，QLDB 会使用 RetireGrant 该操作来删除授权。

创建的授权特定于表。CreateGrant 请求CreateGrant的委托人是创建了表的用户。

记录 CreateGrant 操作的事件与以下示例事件类似。参数包括表的 CMK 的 Amazon 资源名称（ARN）、被授权委托人和停用委托人（ 服务）以及授权涵盖的操作。

```
{
    "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

当您指定客户托管密钥来保护分类账时，QLDB 会创建一个唯一的数据密钥。它向发送GenerateDataKey请求 AWS KMS ，指定账本的客户托管密钥。

记录 GenerateDataKey 操作的事件与以下示例事件类似。该用户是 QLDB 服务账户。这些参数包括客户管理的密钥的 ARN、一个要求为 32 字节长度的数据密钥说明符，以及标识内部密钥层次节点的加密上下文。

```
{
    "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

当您访问分类账时，QLDB 会调用 `Decrypt` 操作来解密分类账存储的数据密钥，以便它可以访问分类账中的加密数据。

记录 `Decrypt` 操作的事件与以下示例事件类似。该用户是 QLDB 服务账户。这些参数包括客户托管密钥的 ARN 和标识内部密钥层次结构节点的加密上下文。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:56Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "key-hierarchy-node-id": "LY4HWMnkeZWKYi6MlitVJC",
            "key-hierarchy-node-version": "1"
        }
    },
    "responseElements": null,
    "requestID": "28f2dd18-3cc1-4fe2-82f7-5154f4933ebf",
    "eventID": "603ad5d4-4744-4505-9c21-bd4a6cbd4b20",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "7b6ce3e3-a764-42ec-8f90-5418c97ec411"
}
```

Encrypt

QLDB 会调用 `Encrypt` 该操作使用您的客户托管密钥将明文加密为密文。

记录 `Encrypt` 操作的事件与以下示例事件类似。该用户是 QLDB 服务账户。这些参数包括客户托管密钥的 ARN 和指定分类账内部唯一 ID 的加密上下文。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "qldb.amazonaws.com"
```

```
    },
    "eventTime": "2021-06-04T21:40:01Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "LedgerId": "F6qRNziJLUXA4Vy2ZUv8YY"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "b2daca7d-4606-4302-a2d7-5b3c8d30c64d",
    "eventID": "b8aace05-2e37-4fed-ae6f-a45a1c6098df",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "ce420ab0-288e-4b4f-aee8-541e18a28aa5"
}
```

DescribeKey

QLDB 会调用 `DescribeKey` 操作来确定您指定的 KMS 密钥是否存在于 AWS 账户 和区域中。

记录 `DescribeKey` 操作的事件与以下示例事件类似。委托 AWS 账户 人是您中指定 KMS 密钥的用户。参数包括客户自主管理型密钥的 ARN。

```
{
    "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"
}
```

Amazon QLDB 中的传输加密

Amazon QLDB 仅接受使用 HTTPS 协议的安全连接，通过安全套接字层（SSL）/传输层安全性协议（TLS）保护网络流量。传输中加密可在数据进出 QLDB 时对其进行加密，为数据提供额外一层数据保护。组织策略、行业或政府法规以及合规性需求通常需要使用静态加密，以便在通过网络传输数据时提高应用程序的数据安全性。

QLDB 还在选定区域提供 FIPS 端点。与标准 AWS 端点不同，FIPS 端点使用符合美国联邦信息处理标准（FIPS）140-2 的 TLS 软件库。与美国政府有业务来往的企业可能需要使用这些端点。有关更多信息，请参阅 https://docs.aws.amazon.com/general/latest/gr/rande.html#FIPS-endpoints 中的AWS 一般参考 FIPS 端点。有关可用于 QLDB 的区域和端点的完整列表，请参阅 Amazon QLDB 端点和限额。

适用于 Amazon QLDB 的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制可以通过身份验证（登录）和授权（具有权限）使用资源的人员。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon MQ 如何与 IAM 协同工作
- 请参阅《Amazon QLDB 开发人员》中的标准权限模式入门
- Amazon QLDB 基于身份的策略示例
- 防止跨服务混淆座席
- AWS 亚马逊 QLDB 的托管策略

- Amazon QLDB 身份和访问问题排查

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 QLDB 中所做的工作。

服务用户 – 如果使用 ACM 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 QLDB 特征来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 QLDB 中的特征，请参阅 Amazon QLDB 身份和访问问题排查。

服务管理员 – 如果您在公司负责管理 资源，则您可能具有 的完全访问权限。您有责任确定您的服务用户应访问哪些 特征和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 搭配使用的更多信息，请参阅Amazon MQ 如何与 IAM 协同工作。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon QLDB 基于身份的策略示例 基于身份的策略示例，请参阅 。

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建帐户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或

AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》 中的访问控制列表（ACL）概览。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon MQ 如何与 IAM 协同工作

在使用 IAM 管理对 QLDB 的访问之前，您应该了解哪些 IAM 功能可用于 QLDB。

将 IAM 功能与 Amazon QLDB 一起使用

IAM 功能	QLDB 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACL	否
ABAC（策略中的标签）	是
临时凭证	是
主体权限	否
服务角色	是
服务相关角色	否

要全面了解 QLDB AWS 服务 和其他功能如何与大多数 IAM 功能配合使用，AWS 服务 请在 IAM 用户指南中查看如何与 IAM 配合使用。

ACM 的基于身份的策略

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南中的创建 IAM 策略。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 QLDB 的基于身份的策略示例

要查看 QLDB 基于身份的策略的示例，请参阅 Amazon QLDB 基于身份的策略示例。

QLDB 内基于资源的策略

支持基于资源的策略	否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

适用于 QLDB 的策略操作

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

有关 Amazon QLDB 操作的列表，请参阅《服务授权参考》中的 Amazon QLDB 定义的操作。

中的策略操作在操作前使用以下前缀：

```
qldb
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 Describe 开头的所有操作，包括以下操作：

```
"Action": "qldb:Describe*"
```

要通过在分类账上运行 PartiQL 语句与 QLDB 事务数据 API（QLDB 会话）交互，您必须按照以下方式授予 SendCommand 操作的权限。

```
"Action": "qldb:SendCommand"
```

有关 STANDARD 权限模式下的分类账，请参阅 PartiQL 权限参考，了解每个 PartiQL 命令所需的其他权限。

要查看 QLDB 基于身份的策略的示例，请参阅 Amazon QLDB 基于身份的策略示例。

QLDB 的策略资源

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

有关 QLDB 资源类型及其 ARN 的列表，请参阅 服务授权参考 中的 Amazon QLDB 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon QLDB 定义的操作。

在 QLDB 中，主要资源是分类账。QLDB 还支持其他资源类型：表和流。但是，只能在现有的分类账范围内创建索引和流。

QLDB 表是分类账日记账中无序文档修订集合的实体化视图。在分类账的 STANDARD 权限模式下，您必须创建 IAM policy 来授予在此表资源上运行 PartiQL 语句的权限。凭借对表资源的权限，您可以运行访问表当前状态的语句。您也可以使用内置 history() 函数查询表的修订历史记录。要了解更多信息，请参阅 请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

> **Note**
>
> 该 CREATE TABLE 语句创建一个具有唯一 ID 和所提供的表名的表。提供的表名称在所有活动表中必须是唯一的。但是，QLDB 允许您停用表，因此可能会有多个非活动表共享相同的表名。因此，表资源 ARN 指的是系统分配的唯一 ID，而不是用户定义的表名。

每个分类账还提供系统定义的目录资源，您可通过查询该资源列出分类账中的所有表和索引。有关 QLDB 数据对象模型的详细信息，请参阅 Amazon QLDB 中的核心概念和术语。

资源具有与其关联的唯一 ARN，如下表所示。

资源类型	ARN
ledger	arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}
table	arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}

资源类型	ARN
catalog	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables`
stream	`arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}`

例如，要在语句中指定 `myExampleLedger` 事件，请使用以下 ARN。

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

要查看 QLDB 基于身份的策略的示例，请参阅 Amazon QLDB 基于身份的策略示例。

QLDB 的策略条件键

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 QLDB 条件键的列表，请参阅《服务授权参考》中的 Amazon QLDB 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon QLDB 定义的操作。

`PartiQLDropIndex` 和 `PartiQLDropTable` 操作支持 `qldb:Purge` 条件键。该条件键根据 PartiQL `DROP` 语句中指定的`purge`值过滤访问。但是，QLDB 目前仅支持 `DROP INDEX` 语句的 `purge = true` 以及 `DROP TABLE` 语句的 `purge = false`。其他 QLDB 操作支持某些全局条件键。

要查看 QLDB 基于身份的策略的示例，请参阅 Amazon QLDB 基于身份的策略示例。

QLDB 中的访问控制列表（ACL）

支持 ACL	否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

QLDB 基于属性的访问控制（ABAC）

支持 ABAC（策略中的标签）	是

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 AWS 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问控制（ABAC）。

有关标记 QLDB 资源的更多信息，请参阅 为 Amazon QLDB 资源贴标签。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅根据标签更新 QLDB 分类账。

将临时凭证用于 QLDB

支持临时凭证	是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

的跨服务主体权限

支持转发访问会话（FAS）	否

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

ACM 的服务角色

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 ACM 的功能。仅当 ACM 提供相关指导时才编辑服务角色。

QLDB 支持用于 `ExportJournalToS3` 和 `StreamJournalToKinesis` API 操作的服务角色，如下一节所述。

在 QLDB 中选择 IAM 角色

在 QLDB 中导出或流式传输日记账区块时，您必须选择一个角色，以允许 QLDB 代表您将对象写入指定的目标。如果您之前创建了一个服务角色，QLDB 会为您提供一个角色列表供您选择。请务必选择一个角色，该角色允许访问您指定的 Amazon S3 存储桶进行导出，或者允许访问您指定的 Amazon Kinesis Data Streams 资源进行数据流。有关更多信息，请参阅 QLDB 中的日记账导出权限 或 QLDB 中的流权限。

> **Note**
>
> 要在请求日记账流时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 `iam:PassRole` 操作的权限。这是为了执行 QLDB 分类账资源上的 `qldb:ExportJournalToS3` 或 QLDB 流子资源上的 `qldb:StreamJournalToKinesis` 而额外需要的权限。

QLDB 的服务相关角色

支持服务相关角色	否

服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅使用 IAM 的AWS 服务 服务。在表中查找服务相关角色列中包含 Yes 额表。选择 Yes 链接以查看该服务的服务相关角色文档。

请参阅《Amazon QLDB 开发人员》中的标准权限模式入门

使用此部分来开始使用 Amazon QLDB 中的标准权限模式。该部分提供了一个参考表，帮助您编写 QLDB 中 PartiQL 操作和表资源的基于身份的策略 AWS Identity and Access Management （IAM）。它还包括在 IAM 中创建权限策略的 step-by-step 教程，以及在 QLDB 中查找表 ARN 和创建表标签的说明。

主题

STANDARD权限模式

Amazon QLDB 现在支持分类账资源STANDARD权限模式。：（推荐）一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。默认情况下，此模式拒绝所有用户请求在此分类账中的任何表上运行任何 PartiQL 命令。

> **Note**
>
> 以前，分类账唯一可用的权限模式是ALLOW_ALL。ALLOW_ALL模式启用了对分类账的 API 级细粒度访问控制，并继续支持 QLDB 分类账，但不建议使用。此模式允许拥有 SendCommand API 权限的用户在分类账中指定的任何表上运行所有 PartiQL 命令（因此，“全部允许”PartiQL 命令）。
>
> 您可以将现有分类账的权限模式从ALLOW_ALL更改为STANDARD。有关信息，请参阅 迁移至标准权限模式。

要允许在标准模式下使用命令，您必须在 IAM 中为特定的表资源和 PartiQL 操作创建权限策略。这是对分类账的 SendCommand API 权限的补充。为了简化这种模式下的策略，QLDB 为 PartiQL 命令引入了一组 IAM 操作，为 QLDB 表引入了一组 Amazon 资源名称（ARN）。有关 QLDB 数据对象模型的详细信息，请参阅 Amazon QLDB 中的核心概念和术语。

PartiQL 权限参考

下表列出了每个 QLDB PartiQL 命令、您必须授予权限才能执行该命令的相应的 IAM 操作，AWS 以及您可以授予权限的资源。您可以在策略的 `Action` 字段中指定这些操作，并在策略的 `Resource` 字段中指定资源值。

Important

- 向这些 PartiQL 命令授予权限的 IAM policy 仅适用于您的分类账，前提是该分类账已分配 `STANDARD` 权限模式。此类策略不适用于`ALLOW_ALL`权限模式下的分类账。

  要了解在创建或更新分类账时如何指定权限模式，请参阅控制台入门中的 Amazon QLDB 分类账的基本操作 或 第 1 步：创建新分类账。
- 要在分类账上运行任何 PartiQL 命令，您还必须授予分类账资源的 SendCommand API 操作权限。这是对下表中列出的 PartiQL 操作和表资源的补充。有关更多信息，请参阅 正在运行数据事务。

Amazon QLDB partiQL 命令和所需权限

命令	所需权限（IAM 操作）	资源	相关操作
CREATE TABLE	qldb:PartiQLCreateTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*	qldb:TagResource （用于在创建时添加标签）
DROP TABLE	qldb:PartiQLDropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
取消删除表	qldb:PartiQLUndropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

命令	所需权限（IAM 操作）	资源	相关操作
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*	
删除 FROM-REMOVE（适用于整个文档）	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*	
更新 FROM（插入、删除或设置）	qldb:PartiQLUpdate	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
REDACT_REVISION（存储过程）	qldb:PartiQLRedact	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
SELECT FROM table_name	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

命令	所需权限（IAM 操作）	资源	相关操作
SELECT FROM information_schema.user_tables	qldb:PartiQLSelect	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /information_schema/user_tables	
SELECT FROM history (table_name)	qldb:PartiQLHistoryFunction	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

有关授予这些 PartiQL 命令权限的 IAM policy 策略文档的示例，请继续 快速入门教程：创建权限策略 或参阅 Amazon QLDB 基于身份的策略示例。

查找表 ID 和 ARN

您可以使用 AWS Management Console 或通过查询表信息_schema.user_ tables 来查找表 ID。要在控制台上查看表的详细信息或查询此系统目录表，您必须拥有系统目录资源的 SELECT 权限。例如，要查找表的 Vehicle 表 ID，可以运行以下语句。

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

此查询以类似于以下示例的格式返回结果。

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
```

```
    ],
    status: "ACTIVE"
}
```

要授予在表上运行 PartiQL 语句的权限，你需按以下 ARN 格式指定表资源。

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

以下是表 ID 的表 ARN 示例。`Au1EiThbt8s0z9wM26REZN`

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

使用 控制台

您还可以使用 QLDB 控制台创建表。

查找表的 ARN（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 在导航窗格中，选择分类账。
3. 在分类帐列表中，选择要查找其表 ARN 的分类帐名称。
4. 在分类账详情页面的表格选项卡下，找到要查找其 ARN 的表名。要复制 ARN，请选择其旁边的复制图标（ ）。

为表格添加标签

现在您可以标记您的资源。要管理现有表的标签，请使用 AWS Management Console 或 API 操作`TagResource`、`UntagResource`和`ListTagsForResource`。有关更多信息，请参阅 为 Amazon QLDB 资源贴标签。

> **Note**
>
> 表资源不会继承其根分类账资源的标签。
>
> 目前，只有 STANDARD 权限模式分类账支持在创建时对表格进行标记。

您还可以在创建表时使用 QLDB 控制台或在 CREATE TABLE PartiQL 语句中指定表标签来定义表标签。下面的示例创建了一个名为 Vehicle 的表，带有标签 environment=production。

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

在创建时为表添加标签，需要同时访问 qldb:PartiQLCreateTable 和 qldb:TagResource 操作。

通过在创建资源时对其进行标记，无需在创建资源后运行自定义标记脚本。标记表后，您可根据这些标签来控制对表的访问。例如：您只能向具有特定标签的表授予完全访问权限。有关 JSON 策略示例，请参阅 基于表格标签对所有操作的完全访问权限。

使用 控制台

您还可以在创建表格时使用 QLDB 控制台来定义表格标签。

在创建表时标记表（控制台）

1. 登录并打开亚马逊 QLDB 控制台，网址为 https://console.aws.amazon.com/qldb。 AWS Management Console
2. 在导航窗格中，选择分类账。
3. 在分类帐列表中，选择要在其中创建表格的分类帐名称。
4. 在分类账详细信息页面的表格选项卡下，选择创建表格。
5. 在创建 DynamoDB 表页面，执行以下操作：

   - 表名称-输入表名称。
   - 标签 — 以键值对形式附加标签来向表添加元数据。Tags（标签）– 您可以将标签添加到任务，帮助您组织和识别它们。

     选择添加标签，然后根据需要输入任何键值对。
6. 根据需要进行设置后，选择 Create table（创建表）。

快速入门教程：创建权限策略

这个教程将引导您完成在 IAM 中为 Amazon QLDB 分类账创建权限策略的步骤，该分类账处于 STANDARD 权限模式。然后，您可以为您的用户、组或角色分配权限。

有关授予 PartiQL 命令和表资源权限的 IAM policy 策略文档的更多示例，请参阅 Amazon QLDB 基于身份的策略示例。

主题

先决条件

在开始之前，请务必执行以下操作：

1. 如果您尚未执行此操作访问 Amazon QLDB，请按照中的 AWS 设置说明进行操作。这些步骤包括注册 AWS 和创建管理用户。
2. 创建新分类账并为该分类账选择 STANDARD 权限模式。要了解操作方法，请参阅控制台入门中的第 1 步：创建新分类账 或 Amazon QLDB 分类账的基本操作。

创建只读策略

要使用 JSON 策略编辑器在标准权限模式下为分类账中的所有表格创建只读策略，请执行以下操作：

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在左侧的导航栏中，选择 Policies（策略）。

   如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择开始使用。
3. 在页面的顶部，选择 Create Policy（创建策略）。
4. 请选择 JSON 选项卡。
5. 对于 Policy Document，复制并粘贴以下文本。此示例策略授予对分类账中所有表的只读访问权限。

   要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

```
{
  "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. 选择Review policy（查看策略）。

> **Note**
>
> 您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy（查看策略），IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在 Review policy（查看策略）页面上，为创建的策略输入 Name（名称）和 Description（说明）（可选）。查看策略摘要以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。

创建完全访问策略

要在标准权限模式下为 QLDB 分类账中的所有表创建完全访问策略，请执行以下操作：

- 使用以下策略文件重复前面的步骤。这个示例策略通过使用通配符（*）覆盖分类账下的所有 PartiQL 操作和所有资源，授予对分类账中所有表的所有 PartiQL 命令的访问权。

> **Warning**
>
> 这是使用通配符（*）允许全部 PartiQL 操作的示例，包括对 QLDB 分类账中的所有表的管理和读/写操作。相反，最佳实践是明确指定要授予的每个操作以及仅指定该用户、角色或组需要的操作。

要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

为特定表创建只读策略

要在标准权限模式下为 QLDB 分类账中的特定表创建只读访问策略，请执行以下操作：

1. 使用 AWS Management Console 或通过查询系统目录表来查找该表的 ARN。`information_schema.user_tables`有关说明，请参阅 查找表 ID 和 ARN。

2. 使用表 ARN 创建策略，允许对表进行只读访问。为此，请使用以下策略文档重复前面的步骤。

   此示例策略仅授予对指定表的只读访问权限。在此示例中，ID 为Au1EiThbt8s0z9wM26REZN。*#################### us-east-1 # 123456789012 # Au1 8s0z9wm26Rez myExampleLedgern#EiThbt*

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

分配权限

创建 QLDB 权限策略后，您可以按如下方式分配权限。

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。

- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

Amazon QLDB 基于身份的策略示例

默认情况下，用户和角色没有创建或修改 资源的权限。他们也无法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的 创建 IAM policy。

有关 QLDB 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅《服务授权参考》 中的 Amazon QLDB 的操作、资源和条件密钥。

目录

    - 允许访问基于请求标签创建表
- 将快照导出到 Amazon S3 存储桶
- 将日记账流式传输到 Kinesis Data Streams
- 根据标签更新 QLDB 分类账

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 QLDB 控制台

要访问 Amazon MQ 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 QLDB 资源的详细信息。 AWS 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色拥有对 QLDB 控制台及其所有功能的完全访问权限，请将 AWS 以下托管策略附加到实体。有关更多信息，请参阅 IAM 用户指南 中的 AWS 亚马逊 QLDB 的托管策略 和 为用户添加权限。

```
AmazonQLDBConsoleFullAccess
```

查询历史权限

除了 QLDB 权限外，某些控制台功能还需要 Database Query Metadata Service 的权限（服务前缀：dbqms）。这是一项仅限内部访问的服务，可在 QLDB 和其他 AWS 服务的控制台查询编辑器上管理您最近和保存的查询。有关DBQMS API操作的完整列表，请参阅 Service Authorization Reference 中的 Database Query Metadata Service。

要允许查询历史记录权限，您可以使用 AWS 托管策略 AmazonQLDB ConsoleFullAccess。此策略使用通配符（dbqms:*）允许对所有资源执行所有 DBQMS 操作。

或者，您也可以创建自定义 IAM policy 并添加以下 DBQMS 操作。QLDB 控制台上的 PartiQL 查询编辑器需要具备使用这些操作的权限以支持查询历史功能。

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

没有查询历史记录的完全访问控制台权限

要允许在 QLDB 控制台上完全访问而无需任何查询历史权限，您可以创建一个自定义 IAM policy，将所有 DBQMS 操作排除在外。例如，以下策略文档允许的权限与 AWS 托管策略 AmazonQLDB 授予的权限相同ConsoleFullAccess，但以服务前缀开头的操作除外。dbqms

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
```

```
        "qldb:CreateLedger",
        "qldb:UpdateLedger",
        "qldb:UpdateLedgerPermissionsMode",
        "qldb:DeleteLedger",
        "qldb:ListLedgers",
        "qldb:DescribeLedger",
        "qldb:ExportJournalToS3",
        "qldb:ListJournalS3Exports",
        "qldb:ListJournalS3ExportsForLedger",
        "qldb:DescribeJournalS3Export",
        "qldb:CancelJournalKinesisStream",
        "qldb:DescribeJournalKinesisStream",
        "qldb:ListJournalKinesisStreamsForLedger",
        "qldb:StreamJournalToKinesis",
        "qldb:GetBlock",
        "qldb:GetDigest",
        "qldb:GetRevision",
        "qldb:TagResource",
        "qldb:UntagResource",
        "qldb:ListTagsForResource",
        "qldb:SendCommand",
        "qldb:ExecuteStatement",
        "qldb:ShowCatalog",
        "qldb:InsertSampleData",
        "qldb:PartiQLCreateIndex",
        "qldb:PartiQLDropIndex",
        "qldb:PartiQLCreateTable",
        "qldb:PartiQLDropTable",
        "qldb:PartiQLUndropTable",
        "qldb:PartiQLDelete",
        "qldb:PartiQLInsert",
        "qldb:PartiQLUpdate",
        "qldb:PartiQLSelect",
        "qldb:PartiQLHistoryFunction"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "kinesis:ListStreams",
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
```

```
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "qldb.amazonaws.com"
        }
      }
    }
  ]
}
```

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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": "*"
        }
    ]
}
```

正在运行数据事务

要通过在分类账上运行 PartiQL 语句与 QLDB 事务数据 API（QLDB 会话）交互，您必须按照以下方式授予 SendCommand API 操作的权限。以下 JSON 文档是一个策略示例，该策略仅向分类账 SendCommand 上的 myExampleLedger API 操作授予权限。

要使用此政策，请将 us-ea *st-1* *#* 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
}
```

如果 myExampleLedger 使用 ALLOW_ALL 权限模式，则此策略授予在分类账中的任何表上运行所有 PartiQL 命令的权限。

您也可以使用 AWS 托管策略授予对所有 QLDB 资源的完全访问权限。有关更多信息，请参阅 AWS 亚马逊 QLDB 的托管策略。

PartiQL 操作和表资源的标准权限

对于 STANDARD 权限模式下的分类账，您可以参考以下 IAM policy 文档作为授予相应的 PartiQL 权限的示例。有关每个 PartiQL 命令所需权限的列表，请参阅 PartiQL 权限参考。

主题

允许对所有 操作的完全访问

以下 JSON 策略文档授予对 myExampleLedger 中的 所有 表使用所有 PartiQL 命令的完全访问权限。此策略产生的效果与对分类账使用 ALLOW_ALL 权限模式相同。

要使用此政策，请将 us-ea *st-1* *#* 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

基于表格标签对所有操作的完全访问权限

以下 JSON 策略文档授予对 myExampleLedger 中的 所有 表使用基于表资源标签的条件来授予 所有 PartiQL 命令的完全访问权限。只有当表标签 environment 具有值 development 时，才会授予权限。

> **Warning**
>
> 这是使用通配符（*）允许全部 PartiQL 操作的示例，包括对 QLDB 分类账中的所有表的管理和读/写操作。相反，最佳实践是明确指定要授予的每个操作以及仅指定该用户、角色或组需要的操作。

要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

读/写入权限

以下 JSON 策略文档授予选择、插入、更新和删除 myExampleLedger 中所有表的数据的权限。此策略不授予密文数据或更改架构的权限，例如，创建和删除表和索引。

> **Note**
>
> 一个 UPDATE 语句需要对正在修改的表执行的 qldb:PartiQLUpdate 和 qldb:PartiQLSelect 操作都具有权限。运行 UPDATE 语句时，除了更新操作外，它还会执行读取操作。要求这两个操作可以确保只有被允许读取表格内容的用户才会被授予 UPDATE 权限。
>
> 同样，一个 DELETE 语句需要同时具有 qldb:PartiQLDelete 和 qldb:PartiQLSelect 操作的权限。

要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

只读访问权限

以下 JSON 策略文档授予对 myExampleLedger 中所有表的只读权限。要使用此政策，请将 us-ea *st-1 #* 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

```
{
    "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"
```

```
            ]
        }
    ]
}
```

对特定表的只读访问

以下 JSON 策略文档授予对 myExampleLedger 中特定表的只读权限。在此示例中，ID 为Au1EiThbt8s0z9wM26REZN。

###################### us-east-1 # 123456789012 # Au1 8s0z9wm26Rez myExampleLedgern#EiThbt

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

允许访问创建表

以下 JSON 策略文档授予在 myExampleLedger 中创建表的权限。qldb:PartiQLCreateTable 操作需要对表资源类型的权限。但是，在运行一个 CREATE TABLE 语句时，新表的表 ID 尚不清楚。因此，授予 qldb:PartiQLCreateTable 权限的策略必须在表 ARN 中使用通配符（*）来指定资源。

要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

```
{
    "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/*"
            ]
        }
    ]
}
```

允许访问基于请求标签创建表

以下 JSON 策略文档使用基于 aws:RequestTag 上下文密钥的条件来授予在中创建表的权限 myExampleLedger。只有当请求标签 environment 具有值时，才会授予权限 development。在创建时为表添加标签，需要同时访问 qldb:PartiQLCreateTable 和 qldb:TagResource 操作。要了解如何在创建表时标记表，请参阅 为表格添加标签。

要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

将快照导出到 Amazon S3 存储桶

步骤 1：QLDB 日记账导出权限

在以下示例中，您授予用户对 QLDB 账本资源执行`qldb:ExportJournalToS3`操作的 AWS 账户 权限。您还可以授予对要传递给 QLDB 服务的 IAM 角色资源执行 `iam:PassRole` 操作的权限。这是所有日记账流请求所必需的。

############ us-ea st-1#123456789012 myExampleLedger# ql d b-s3-export ### #######

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

步骤 3：设置 Amazon S3 存储桶上的权限

在以下示例中，您使用 IAM 角色向 QLDB 授予写入您的 Amazon S3 存储桶 DOC-EXAMPLE-BUCKET 的访问权限。所有 QLDB 日记账导出也是必需的。

除了授予 s3:PutObject 权限外，该策略还授予 s3:PutObjectAcl 权限以设置对对象的访问控制列表（ACL）权限。

要使用此策略，请将 DOC-EXAMPLE-BUCKET 替换为 Amazon S3 存储桶的名称。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

然后，将此权限策略附加到 QLDB 可代入您的 Amazon S3 存储桶的 IAM 角色。以下 JSON 文档是信任策略的示例，该策略仅允许 QLDB 为账户 123456789012 中的任何 QLDB 资源担任 IAM 角色。

要使用此策略，请将示例中的 *us-east-1* 和 *123456789012* 12 替换为您自己的信息。

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

将日记账流式传输到 Kinesis Data Streams

第 1 步：QLDB 日记账流权限

在以下示例中，您授予用户对 AWS 账户 账本中的所有 QLDB 流子资源执行`qldb:StreamJournalToKinesis`操作的权限。您还可以授予对要传递给 QLDB 服务的 IAM 角色资源执行 `iam:PassRole` 操作的权限。这是所有日记账流请求所必需的。

要使用此政策，请使用您自己的信息替换*####* us-east-1 *#* 1234567890 *myExampleLedger*12 和。*qldb-kinesis-stream*

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

第 2 步：Kinesis Data Streams 权限

在以下示例中，您使用 IAM 角色向 QLDB 授予向您的 Amazon Kinesis 数据流写入数据记录的权限。*stream-for-qldb*这是所有日记账流请求所必需的。

要使用此政策，请将 us-ea *st-1 #* 123456789012 和示例中的信息替换为你自己的信息。*stream-for-qldb*

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

然后，您将此权限策略附加到 QLDB 可以代入的 IAM 角色来访问您的 Kinesis 数据流。以下 JSON 文档是信任策略的示例，该策略允许 QLDB 仅为分类账 myExampleLedger 账户 123456789012 中的任何 QLDB 流担任 IAM 角色。

要使用此政策，请将 us-ea *st-1 #* 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

根据标签更新 QLDB 分类账

您可以在基于身份的策略中使用条件，以便基于标签控制对 资源的访问。该示例说明了如何创建策略以允许查看队组。不过，只有在事物标签 Owner 具有该用户的用户名值时，才会授予权限。此策略还授予在控制台上完成此操作的必要权限。

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

```
    ]
}
```

您可以将此策略附加到您账户中的用户。如果名为 richard-roe 的用户尝试更新 QLDB 分类账，则必须对该分类账进行标记 Owner=richard-roe 或 owner=richard-roe。否则，他将被拒绝访问。条件标签键 Owner 匹配 Owner 和 owner，因为条件键名称不区分大小写。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。

一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止出现混乱的代理问题，我们 AWS 提供了一些工具，可帮助您保护所有服务的数据，这些服务委托人已被授予访问您账户中资源的权限。

我们建议在资源策略中使用 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键，以限制 Amazon QLDB 为其他服务提供的资源访问权限。如果使用两个全局条件上下文键，在同一策略语句中使用时，aws:SourceAccount 值和 aws:SourceArn 值中的账户必须使用相同的账户 ID。

下表列出了 ExportJournalToS3 和 StreamsJournalToKinesis QLDB API 操作的可能值aws:SourceArn。这些操作在此安全问题范围内，因为它们调用 AWS Security Token Service (AWS STS) 来代入您指定的 IAM 角色。

API 操作	调用的服务	aws：SourceArn
ExportJournalToS3	AWS STS (AssumeRole)	允许 QLDB 担任账户中任何 QLDB 资源的角色： `arn:aws:qldb: us-east-1 :123456789012 :*` 目前，QLDB 仅支持该通配符 ARN 用于日记账导出。
StreamsJournalToKinesis	AWS STS (AssumeRole)	允许 QLDB 为特定 QLDB 流担任角色：

API 操作	调用的服务	aws：SourceArn

		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		
		注意：只有在创建流资源后，您才能在 ARN 中指定流 ID。使用此 ARN，您可以允许该角色仅用于单个 QLDB 流。
		
		允许 QLDB 担任分类账中任何 QLDB 流的角色：
		
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`
		
		允许 QLDB 担任账户中任何 QLDB 流的角色：
		
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		
		允许 QLDB 担任账户中任何 QLDB 资源的角色：
		
		`arn:aws:qldb: us-east-1 :123456789012 :*`

防范混淆代理问题最有效的方法是使用 aws:SourceArn 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符（*）的 aws:SourceArn 全局上下文条件键，例如arn:aws:qldb:us-east-1:*123456789012*:*。

以下关于 IAM 角色的示例诚信角色演示如何使用 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键来防范混淆代理问题。使用此信任策略，QLDB 只能为分类账 123456789012 账户 myExampleLedger 中的任何 QLDB 流扮演角色。

要使用此政策，请将 us-ea *st-1 #* 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

```
{
  "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 亚马逊 QLDB 的托管策略

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

有关 AWS 这些托管策略中的 QLDB API 操作的更多信息，请参阅。Amazon QLDB API 参考

主题

- AWS 托管策略：亚马逊 QLDB ReadOnly
- AWS 托管策略：亚马逊 QLDB FullAccess
- AWS 托管策略：亚马逊 QLDB ConsoleFullAccess
- QLDB 对托管策略的更新 AWS

AWS 托管策略：亚马逊 QLDB ReadOnly

使用 AmazonQLDB ReadOnly 策略向所有 QLDB 资源授予只读权限。您可以将 策略附加得到 IAM 身份。

权限详细信息

此策略包含以下qldb服务权限。

- 允许主体描述和列出所有 QLDB 资源及其标签。这些资源包括分类账、Amazon S3 导出作业和流向 Kinesis Data Streams 的流。
- 允许委托人从任何分类账的日记账中获取区块、摘要或修订，以加密方式验证数据。
- 不允许主体对任何分类账中的任何表运行任何 PartiQL 命令。

AWS 托管策略：亚马逊 QLDB FullAccess

使用 AmazonQLDB FullAccess 策略通过 QLDB API 或. 向所有 QLDB 资源授予完全管理权限。 AWS CLI您可以将 策略附加得到 IAM 身份。

权限详细信息

此策略包含以下权限。

- qldb
  - 允许委托人创建、描述、列出和管理所有 QLDB 资源及其标签。这些资源包括分类账、Amazon S3 导出作业和流向 Kinesis Data Streams 的流。
  - 允许主体使用 QLDB 驱动程序或 QLDB Shell 在任何分类账上运行所有 PartiQL 命令。
  - 允许委托人从任何分类账的日记账中获取区块、摘要或修订，以加密方式验证数据。
- iam— 允许委托人将您账户中的任何 IAM 角色资源传递给 QLDB 服务。这是所有日记账流请求所必需的。

AWS 托管策略：亚马逊 QLDB ConsoleFullAccess

使用 AmazonQLDB ConsoleFullAccess 政策，通过、 AWS Management Console QLDB API 或，向所有 QLDB 资源授予完全管理权限。 AWS CLI您可以将 策略附加得到 IAM 身份。

权限详细信息

此策略包含以下权限。

- `qldb`
  - 允许委托人创建、描述、列出和管理所有 QLDB 资源及其标签。这些资源包括分类账、Amazon S3 导出作业和流向 Kinesis Data Streams 的流。
  - 允许主体使用QLDB 控制台、QLDB 驱动程序或 QLDB Shell 在任何分类账上运行所有 PartiQL 命令。
  - 允许委托人使用 QLDB 控制台在任何分类账中插入示例应用程序数据。
  - 允许委托人从任何分类账的日记账中获取区块、摘要或修订，以加密方式验证数据。
- `dbqms`— 允许委托人使用 Database Query Metadata Service 中的所有操作。这是一项仅限内部使用的服务，QLDB 控制台需要该服务，用来为 PartiQL 查询编辑器创建、描述和管理最近和保存的查询。
- `kinesis`— 允许委托人描述和列出 Amazon Kinesis Data Streams 资源。这些资源是 QLDB 流资源可以向其写入数据的目标目标。
- `iam`— 允许委托人将您账户中的任何 IAM 角色资源传递给 QLDB 服务。这是所有日记账流请求所必需的。

QLDB 对托管策略的更新 AWS

查看自 QLDB AWS 托管策略开始跟踪这些更改以来该服务更新的详细信息。要获得有关此页面更改的自动提示，请订阅 ??? 页面上的 RSS 馈送。

更改	描述	日期
亚马逊 qldb FullAccess、AmazonQLD ConsoleFullAccess B — 现有政策的更新	QLDB 添加了一项新权限，允许委托人在权限模式下编辑所有分类账中的文档修订。`STANDARD`	2022 年 11 月 4 日
亚马逊 qldb FullAccess、AmazonQLD ConsoleFullAccess B — 现有政策的更新	QLDB 添加了新的权限，允许委托人将您账户中的任何 IAM 角色资源传递给 QLDB 服务。这是所有日记账流请求所必需的。	2021 年 9 月 2 日

更改	描述	日期
AmazonQldb ReadOnly — 更新现有政策	QLDB 删除了之前列出两次的 `qldb:GetBlock` 重复操作，并对 `"Effect"` 字段进行了重新排序，使其显示在 `"Action"` 字段之前。	2021 年 7 月 1 日
亚马逊 qldb FullAccess、AmazonQLD ConsoleFullAccess B — 现有政策的更新	QLDB 添加了新的权限，允许委托人更新所有分类账中的权限模式，并在新的权限模式下在所有分类账中运行所有 PartiQL 命令。`STANDARD` `STANDARD` 权限模式支持 PartiQL 命令的表级访问控制和粒度。为了简化新的权限模式，QLDB 为 PartiQL 命令类型引入了一组 IAM 操作，为 QLDB 表资源引入了一组 Amazon 资源名称（ARN）。这两项策略已更新，加入了新的 PartiQL 操作，从而授予对 `STANDARD` 分类账的完全访问权限。	2021 年 5 月 27 日
IPAM 已开启跟踪更改	QLDB 开始跟踪其托管策略的更改。 AWS	2021 年 3 月 1 日

Amazon QLDB 身份和访问问题排查

使用以下信息可帮助您诊断和修复在使用 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 中执行操作
- 我无权执行 iam：PassRole
- 我想允许我以外的人访问我的 AWS 账户 QLDB 资源

我无权在 中执行操作

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是向您提供登录凭证的人。

当 mateojackson 用户尝试使用控制台查看有关虚构 *myExampleLedger* 资源的详细信息，但不拥有虚构 qldb:*DescribeLedger* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 qldb:*DescribeLedger* 操作访问 *myExampleLedger* 资源。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 的 IAM 用户尝试使用控制台在 marymajor 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。管理员是向您提供登录凭证的人。

有关该错误的疑难解答指南，该错误与日记账导出或流操作有关，请参阅 Amazon QLDB 故障排除。

我想允许我以外的人访问我的 AWS 账户 QLDB 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 QLDB 是否支持这些特征，请参阅 Amazon MQ 如何与 IAM 协同工作。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

Amazon S3 中的日记账记录和监控

监控是维护 Amazon QLDB 和您的解决方案的可靠性、可用性和性能的重要组成部分。 AWS 您应该从 AWS 解决方案的所有部分收集监控数据，以便在出现多点故障时可以更轻松地进行调试。不过，在开始监控 之前，您应制定一个监控计划并在计划中回答下列问题：

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

下一步，通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 性能的基准。在监控时，存储历史监控数据，以便将此数据与当前性能数据进行比较，确定正常性能模式和性能异常，并设计解决问题的方法。

要建立基准，至少应监控以下项目：

- 读取和写入 I/O 和存储，以便您可以出于计费目的跟踪分类账的消费模式。
- 命令延迟，这样您就可以在运行数据操作时跟踪分类账的性能。
- 系统错误，以便您可以确定是否有任何请求导致了错误。

主题

监控工具

AWS 提供了可用于监控 Amazon QLDB 的各种工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

主题

自动监控工具

您可以使用以下自动化监控工具来监控 并在出现错误时报告：

- A CloudWatch mazon Alarms — 在您指定的时间段内观察单个指标，并根据该指标在多个时间段内相对于给定阈值的值执行一项或多项操作。该操作是发送到亚马逊简单通知服务 (Amazon SNS) Simple Notification Scaling 主题或亚马逊 EC2 Auto Scaling 策略的通知。 CloudWatch 警报不会仅仅因为它们处于特定状态就调用操作；该状态必须已更改并保持了指定的时间段。有关更多信息，请参阅 使用 Amazon 进行监控 CloudWatch。
- Amazon CloudWatch Logs — 监控、存储和访问来自 AWS CloudTrail 或其他来源的日志文件。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的监控日志文件。
- Amazon CloudWatch Events — 匹配事件并将其路由到一个或多个目标函数或流，以进行更改、捕获状态信息并采取纠正措施。有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的什么是亚马逊 CloudWatch 活动。
- AWS CloudTrail 日志监控-在账户之间共享日志文件，通过将 CloudTrail 日志文件发送到 “日志” 来实时监控 CloudWatch 日志文件，用 Java 编写日志处理应用程序，并验证您的日志文件在传送后是否未更改 CloudTrail。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 CloudTrail 日志文件。

手动监控工具

监控 QLDB 的另一个重要部分是手动监控警报未涵盖 CloudWatch 的项目。QLDB CloudWatch、Trusted Advisor、 AWS Management Console 和其他仪表板提供了环境状态 at-a-glance 的视图。AWS 建议还查看 Amazon DynamoDB 的日志文件。

- 控制面板显示以下信息：
  - 读取和写入 I/O
  - 日记账和索引存储
  - 命令延迟
  - 异常
- CloudWatch 主页显示以下内容：
  - 当前告警和状态
  - 告警和资源图表
  - 服务运行状况

  此外，您还可以使用 CloudWatch 执行以下操作：
  - 创建自定义控制面板以监控您关心的服务
  - 绘制指标数据图，以排除问题并弄清楚趋势
  - 搜索和浏览您的所有 AWS 资源指标
  - 创建和编辑告警以接收问题通知

使用 Amazon 进行监控 CloudWatch

您可以使用监控 Amazon QLDB，它会收集 CloudWatch来自 Amazon QLDB 的原始数据并将其处理为可读的指标。 near-real-time 这些统计数据会保存两周，以便您能够访问历史信息，并更好地了解 Web 应用程序或服务的执行情况。默认情况下，QLDB 指标数据会在 1 或 15 分钟内自动发送 CloudWatch 到。有关更多信息，请参阅什么是亚马逊 CloudWatch、亚马逊 CloudWatch 事件和亚马逊 CloudWatch 日志？ 在《亚马逊 CloudWatch 用户指南》中。

主题

- 如何使用 DAX 指标？
- Amazon QLDB 指标与维度
- 创建 CloudWatch 警报以监控 Amazon QLDB

如何使用 DAX 指标？

报告的指标为您提供了可通过不同方式分析的信息。下面的列表显示这些指标的一些常见用途。这些是入门建议，并不全面。

- 可以在指定时间段内监控 JournalStorage 和 IndexedStorage，跟踪分类账消耗的磁盘空间。
- 你可以监控指定的时间段内的 ReadIOs 和 WriteIOs，以追踪您的分类账正在处理多少请求。
- 您可以通过监控 CommandLatency 来跟踪分类账的数据操作性能，并分析导致最大延迟的命令类型。

Amazon QLDB 指标与维度

当您与 Amazon QLDB 互动时，它会将以下指标和维度发送到。 CloudWatch存储指标每 15 分钟报告一次，所有其他指标每分钟汇总和报告一次。您可以使用以下流程查看 QLDB 的指标。

使用 CloudWatch 控制台查看指标

指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 如果需要，可以更改区域。在导航栏上，选择 AWS 资源所在的区域。有关更多信息，请参阅区域和端点。
3. 在导航窗格中，选择 Metrics（指标）。
4. 在 All metrics（全部指标）选项卡下，选择 QLDB。

要查看指标，请使用 AWS CLI

- 在命令提示符处，使用以下命令。

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch 显示了 QLDB 的以下指标。

Amazon QLDB 指标与维度

此处列出了亚马逊 QLDB 发送给 CloudWatch 亚马逊的指标和维度。

ALDB 指标

指标	描述
JournalStorage	分类账日记账使用的磁盘空间总量，每 15 分钟报告一次。该日记账包含所有数据更改的完整、不可变且可验证的历史记录。 单位：Bytes 维度：LedgerName
IndexedStorage	分类账表、索引和索引历史记录使用的磁盘空间总量，每隔 15 分钟报告一次。索引存储包含的分类账数据针对高性能查询进行了优化。 单位：Bytes 维度：LedgerName
ReadIOs	读取 I/O 请求的数量，每隔一分钟报告一次。这会捕获所有类型的读取操作，包括数据事务、验证请求、日记账导出和日记账流。 单位：Count 维度：LedgerName
WriteIOs	每隔一分钟报告的写入 I/O 请求数。 单位：Count 维度：LedgerName
CommandLatency	数据操作所花费的时间，以一分钟为间隔报告。 单位：Milliseconds 维度：CommandType, LedgerName

指标	描述
IsImpaired	指示 Kinesis Data Streams 上的日记账流是否受到影响的标志，以一分钟的间隔报告。值为 1 表示流处于受损状态，否则 0 表示不处于受损状态。 单位：Boolean（0 或 1） 维度：LedgerName, StreamId
OccConflictExceptions	向 QLDB 发出的生成请求的数量。OccConflictException 有关乐观并发控制（OCC）的信息，请参阅。Amazon QLDB 并发模型 单位：Count
Session4xxExceptions	向 QLDB 发出的生成 HTTP 4xx 错误的请求数。 单位：Count
Session5xxExceptions	向 QLDB 发出的生成 HTTP 5xx 错误的请求数。 单位：Count
SessionRateExceededExceptions	向 QLDB 发出的生成请求的数量。SessionRateExceededException 单位：Count

指标的维度

QLDB 的指标是通过账户、账本名称、流 ID 或命令类型的值进行限定的。您可以使用 CloudWatch 控制台按下表中的任何维度检索 QLDB 数据。

维度	描述
LedgerName	此维度将数据限制为特定表。该值可以是当前账本 AWS 区域 和当前账本中的任何账本名称 AWS 账户。

维度	描述
StreamId	此维度将数据限制为特定流标签。此值可以是当前 AWS 区域 和当前账本的任意流 ID AWS 账户。
CommandType	此维度将数据限制为以下 QLDB 数据 API 命令之一： • AbortTransaction • CommitTransaction • EndSession • ExecuteStatement • FetchPage • StartSession • StartTransaction 要了解 QLDB 如何使用这些命令来管理数据操作，请参阅 驱动程序会话管理。

创建 CloudWatch 警报以监控 Amazon QLDB

您可以创建亚马逊警报，在 CloudWatch 警报状态发生变化时发送亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 消息。告警会监控您指定的时间段内的某个指标。它在多个时间段内根据相对于给定阈值的指标值，执行一项或多项操作。操作是一个发送到 Amazon SNS 主题或 Auto Scaling 策略的通知。

警报仅针对持续的状态变化调用操作。 CloudWatch 警报不会仅仅因为它们处于特定状态就调用操作。该状态必须已改变并在指定的若干个时间段内保持不变。

有关创建 CloudWatch 警报的更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。

使用事件自动化 Amazon QLDB CloudWatch

Amazon CloudWatch Events 使您能够自动处理 AWS 服务 并自动响应系统事件，例如应用程序可用性问题或资源更改。来自 AWS 服务 的事件以近乎实时的方式传递到 CloudWatch 事件。您可以编写简单规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。可自动触发的操作包括：

- 调用函数 AWS Lambda
- 调用 Amazon EC2 Run Command
- 将事件中继到 Amazon Kinesis Data Streams
- 激活 AWS Step Functions 状态机
- 通知 Amazon SNS 主题或 Amazon SQS 队列

每当您的账本资源状态发生变化时，Amazon QLDB 就会 CloudWatch 向事件报告事件。 AWS 账户目前，只有QLDB账本资源才 at-least-once 会有保证地发出事件。

以下是 QLDB 报告的事件示例，在该事件中，分类账的状态更改为 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"
    }
}
```

在 QLDB 中使用 CloudWatch 事件的一些示例可能包括但不限于以下内容：

- 在账本最初 CREATING 状态创建并最终变为 ACTIVE 时，激活一个 Lambda 函数。
- 当你的分类账状态变为 DELETING 时，然后再变为 DELETED 时，通知一个 Amazon SNS 主题。

有关更多信息，请参阅 Amazon Ev CloudWatch ents 用户指南。

使用记录亚马逊 QLDB API 调用 AWS CloudTrail

Amazon QLDB AWS CloudTrail与一项服务集成，该服务提供用户、角色或用户在 QLDB 中采取的操作的 AWS 服务 记录。 CloudTrail 将 QLDB 的所有资源管理 API 调用捕获为事件。捕获的调用包括来自 QLDB 控制台的调用和对 API 操作的代码调用。如果您创建了跟踪，则可以允许持续向亚马逊

简单存储服务 (Amazon S3) Storage Service 存储桶传送 CloudTrail 事件，包括 QLDB 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 主机上的事件历史记录中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 QLDB 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，包括如何配置和启用它，请参阅AWS CloudTrail 用户指南。

QLDB 信息在 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当 QLDB 中出现支持的活动时，该活动将 AWS 服务 与其他事件一起记录 CloudTrail 在事件历史记录中。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史查看 CloudTrail 事件。

要持续记录您的事件 AWS 账户，包括 QLDB 的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务 ，以进一步分析和处理 CloudTrail 日志中收集的事件数据。

有关更多信息，请参阅《AWS CloudTrail 用户指南》中的以下主题：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件
- 从多个账户接收 CloudTrail 日志文件

所有 QLDB 资源管理和非交易数据 API 操作均 CloudTrail 由 Amazon QLDB API 参考记录并记录在案。例如，对`CreateLedger`、`DescribeLedger`、和`DeleteLedger`操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是 用户凭证发出的
- 请求是使用角色还是联合用户的临时安全凭证发出的
- 请求是否由其他人提出 AWS 服务

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 &QS; 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

以下示例显示了演示这些操作的 CloudTrail 日志条目：

- `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"
    }
  ]
}
```


Amazon MQ 的合规性验证

作为 AWS 多项合规计划的一部分，第三方审计师评估 Amazon QLDB 的安全与合规性，包括但不限于以下内容：

- 系统和组织控制（SOC）
- 支付卡行业 (PCI)
- 国际标准化组织 (ISO)
- 信息系统安全管理评测制度（ISMAP）
- 健康保险流通与责任法案 (HIPAA)

> **Note**
>
> 这不是 Amazon QLDB 认证的详尽列表。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 提供了您内部安全状态的全面视图 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。

- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

Amazon QLDB 中的恢复功能

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

消息持久性

QLDB 日记账存储功能是在事务提交时同步复制到多个可用区。这样可以确保即使日记账存储的可用区完全出现故障，也不会损害数据完整性或维持有效服务的能力。此外，QLDB 日记还包含容错存储的异步存档。此功能支持在多个可用区同时出现存储故障的可能性极低的情况下进行灾难恢复。

QLDB 索引存储通过复制到多个可用区来支持。这样可以确保即使索引存储的可用区完全出现故障，也不会损害数据完整性或维持有效服务的能力。

数据持久性功能

除了 AWS 全球基础设施外，QLDB 还提供以下功能，以帮助支持您的数据弹性和备份需求。

QLDB 服务功能

按需导出日记账

QLDB 提供按需日记账导出功能。将分类账中的日记账区块导出到 Amazon S3 存储桶，访问日记账内容。您可以将这些数据用于各种目的，例如数据留存、分析和审计。有关更多信息，请参阅从 Amazon QLDB 导出日记账数据。

备份与还原

目前不支持自动恢复导出。导出提供了按您定义的频率创建额外数据冗余的基本功能。然而，还原场景取决于应用程序，其中导出的记录可能通过利用相同或类似的摄取方法写回到一个新的账本。

QLDB 目前不提供专用的备份和相关还原功能。

日记账流

QLDB 还提供连续的日记账流功能。您可以将 QLDB 日记账流与 Amazon Kinesis 流媒体平台集成，以处理实时日记账数据。有关更多信息，请参阅Amazon QLDB 流式传输日记账数据。

QLDB 设计层面

QLDB 旨在抵御逻辑损坏。QLDB 日记账是不可变的，可确保所有已提交的事务都保留在日记账中。此外，每一次提交的文档变更都会被记录下来，因为这样point-in-time可以查看账本数据的任何意外更改。

QLDB 目前不提供针对逻辑损坏情形的自动恢复功能。

Amazon MQ 中的基础设施安全性

作为一项托管服务，Amazon QLDB 受 AWS 《亚马逊网络服务：安全流程概述》白皮书中描述的全球网络安全程序的保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 QLDB。客户端必须支持传输层安全性协议 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密（PFS）的密码套件，例如 Ephemeral Diffie-Hellman（DHE）或 Elliptic Curve Ephemeral Diffie-Hellman（ECDHE）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用与 IAM 委托人关联的编程证书来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

您还可以将 虚拟私有云（VPC）端点用于 QLDB。接口 VPC 端点使 Amazon VPC 资源可以使用其私有 IP 地址访问 Amazon QLDB Auto Scaling，而无需接触公有互联网。有关更多信息，请参阅使用接口端点（AWS PrivateLink）访问 Amazon QLDB。

使用接口端点（AWS PrivateLink）访问 Amazon QLDB

您可以使用 AWS PrivateLink 在您的 VPC 和 Amazon QLDB 之间创建私有连接。您可以像在 VPC 中一样访问 QLDB，无需使用互联网网关、NAT 设备、VPN 连接或连接。 AWS Direct Connect VPC 中的实例不需要公有 IP 地址即可访问 Amazon EKS。

您可以通过创建由 AWS PrivateLink提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管式网络接口，用作发往 Amazon EKS 的流量的入口点。

有关更多信息，请参阅 AWS PrivateLink 指南中的通过 AWS PrivateLink访问 AWS 服务。

主题

- 的注意事项
- 为 创建接口端点
- 为 VPC 端点创建端点策略
- 适用于 QLDB 的 VPC 端点的可用性

的注意事项

在为 Amazon EKS 设置接口端点之前，请首先检查《AWS PrivateLink 指南》中的 注意事项。

> **Note**
>
> QLDB 仅支持通过接口端点调用 QLDB 会话事务数据 API。此 API 仅包含该SendCommand操作。在分类账的 STANDARD 权限模式下，您可以在此 API 中控制对特定 PartiQL 操作的权限。

为 创建接口端点

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface () 为 QLDB 创建接口终端节点。AWS CLI有关更多信息，请参阅 AWS PrivateLink 指南中的创建接口端点。

使用以下服务名称为 Amazon EKS 创建接口端点：

```
com.amazonaws.region.qldb.session
```

如果为接口端点启用私有 DNS，则可使用区域默认 DNS 名称向 Lambda 发出 API 请求，例如 。例如，`session.qldb.us-east-1.amazonaws.com`。

为 VPC 端点创建端点策略

端点策略是一种 IAM 资源，您可以将其附加到接口端点。默认端点策略允许通过接口端点访问 Amazon QLDB API 的完全权限。要控制允许从 VPC 访问 Amazon QLDB API 的权限，请将自定义端点策略附加到接口端点。

端点策略指定以下信息：

- 可以执行操作的主体（ AWS 账户、用户和角色）。

- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅 AWS PrivateLink 指南中的使用端点策略控制对服务的访问权限。

您还可以在附加到用户、组或角色的策略中使用 Condition 字段，以仅允许通过指定的接口端点访问。结合使用时，端点策略和 IAM policy 可以将对指定分类账的特定 QLDB 操作的访问权限限制为指定的接口端点。

示例：限制对特定端点的访问

下面是用于 的 VPC 端点策略的示例。当您将此策略附加到接口端点时，它会向指定分类账资源上的所有委托人授予对 SendCommand 操作和 PartiQL 只读操作的访问权限。在此示例中，分类账必须处于 STANDARD 权限模式。

要使用此政策，请将 us-ea *st-1* # 123456789012 和示例中的信息替换为你自己的信息。*myExampleLedger*

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

```
}
```

IAM policy 示例：仅限制从特定接口端点访问 QLDB 分类账

下面是适用于 QLDB 的 IAM 基于身份的权限策略的示例。当您将此策略附加到用户、角色或组时，它只允许从指定的接口端点SendCommand访问分类账资源。

###################### us-east-1 # 123456789012 # vpce-1a2b3 myExampleLedgerc4d#

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

适用于 QLDB 的 VPC 端点的可用性

Amazon QLDB 支持所有可用的 AWS 区域 中具有策略的接口端点。有关可用区域的完整列表，请参阅 AWS 一般参考 中的 Amazon QLDB 端点和限额。

Amazon QLDB 故障排除

以下部分提供了使用 Amazon QLDB 时可能遇到的常见错误的汇总列表，以及如何排除这些错误的指导。

有关特定于 IAM 访问权限的故障排除指南，请参阅 Amazon QLDB 身份和访问问题排查。

有关调优 PartiQL 语句的最佳实践，请参见 优化查询性能。

主题

使用 QLDB 驱动程序运行事务

本节列出了 Amazon QLDB 驱动程序在分类账上运行 PartiQL 事务时可能返回的常见异常。有关此功能的更多信息，请参阅驱动程序入门。有关配置和使用驱动程序的最佳实践，请参阅 驱动程序推荐。

每个异常都包括特定的错误消息，然后是简短的描述和可能的解决方案建议。

CapacityExceededException

消息：容量已超出

Amazon QLDB 拒绝了该请求，因为它超出了分类账的处理能力。QLDB 对每个分类账强制执行内部扩展限制，以维护服务的运行状况和性能。此限制因每个请求的工作负载大小而异。例如，如果请求执行效率低下的数据事务，例如由非索引限定查询产生的表扫描，则该请求的工作负载可能会增加。

我们建议您在重试请求之前等待。如果您的应用程序一直遇到此异常，请优化您的语句并降低发送到分类账的请求的速率和数量。语句优化的示例包括每个事务运行更少的语句和调优表索引。要了解如何优化语句和避免表扫描，请参阅 优化查询性能。

建议使用最新版本的 QLDB 驱动程序。驱动程序具有默认的重试策略，该策略使用指数回退和抖动来自动重试此类异常。指数回退的原理是对于连续错误响应，重试等待间隔越来越长。

InvalidSessionException

消息：事务 *transactionId* 已过期

事务已超过其最长生命周期。在提交之前，事务最多可运行 30 秒。超过此时间限制后，QLDB 会拒绝在事务完成的任何工作，并丢弃运行该事务会话。此限制通过启动事务（而不提交或取消事务）以保护客户端免遭泄漏会话。

如果这是应用程序中常见的异常，那么很可能是事务运行时间过长。如果事务运行时间超过 30 秒，请优化您的语句以加快事务速度。语句优化的示例包括每个事务运行更少的语句和调优表索引。有关更多信息，请参阅优化查询性能。

InvalidSessionException

消息：会话 *sessionId* 已过期

QLDB 放弃了会话，因为它已超过其最大总生命周期。无论事务是否处于活动状态，QLDB 都会在 13-17 分钟后丢弃会话。会话丢失或受损的原因有很多，如硬件故障、网络故障或应用程序重启。因此，QLDB 对会话强制使用最长生命周期，以确保客户端软件能应对会话故障。

如果您遇到此异常，我们建议您获取一个新的会话并重试该事务。我们还建议使用最新版本的 QLDB 驱动程序，该驱动程序代表应用程序管理会话池及其运行状况。

InvalidSessionException

消息：没有这样的会话

客户试图使用不存在的会话与 QLDB 进行事务处理。假设客户端正在使用先前存在的会话，则由于以下原因之一，该会话可能不再存在：

- 如果会话涉及内部服务器故障（即 HTTP 响应代码 500 错误），那么 QLDB 可能会选择完全放弃该会话，而不是允许客户使用状态不确定的会话进行事务。然后，对该会话的任何重试尝试都会失败，并出现此错误。
- 过期的会话最终会被 QLDB 遗忘。然后，任何继续使用会话的尝试都会导致此错误，而不是初始 `InvalidSessionException`。

如果您遇到此异常，我们建议您获取一个新的会话并重试该事务。我们还建议使用最新版本的 QLDB 驱动程序，该驱动程序代表应用程序管理会话池及其运行状况。

RateExceededException

消息：已超出速率

QLDB 根据调用者的标识对客户端进行节流。QLDB 使用令牌桶节流算法在每个区域、每个账户的基础上强制执行节流。QLDB 这样做是为了帮助提高服务的性能，并确保所有QLDB 客户的公平使用。例如，尝试使用 `StartSessionRequest` 操作获取大量并发会话可能会导致节流。

为了保持应用程序的运行状况并缓解进一步的节流，您可以使用指数回退和抖动来重试此异常。指数回退的原理是对于连续错误响应，重试等待间隔越来越长。建议使用最新版本的 QLDB 驱动程序。驱动程序具有默认的重试策略，该策略使用指数回退和抖动来自动重试此类异常。

如果您的应用程序持续受到 QLDB 的 `StartSessionRequest` 调用节流，那么最新版本的 QLDB 驱动程序也可以提供帮助。驱动程序维护着一个跨事务重复使用的会话池，这有助于减少应用程序发出的 `StartSessionRequest` 调用次数。要请求提高 API 节流限额，请联系 AWS Support 中心。

LimitExceededException

消息：已超出会话限制

分类账的活跃会话数量超过了其配额（也称为限制）。此配额在 Amazon QLDB 资源中的限额和限制 中定义。分类账的活动会话计数最终是一致的，并且始终在配额附近运行的分类账可能会定期看到此异常。

为了保持应用程序的运行状况，我们建议重试此异常。为避免出现这种异常，请确保在所有客户端上为单个分类账配置的并发会话不超过 1,500 个。例如，您可以使用适用于 Java 的 Amazon QLDB 驱动程序的 maxConcurrentTransactions 方法来配置驱动程序实例中的最大可用会话数。

QldbClientException

消息：流结果仅在父事务打开时有效

事务已关闭，无法用于从 QLDB 检索结果。事务在提交或取消时关闭。

当客户端直接处理 `Transaction` 对象，并且在提交或取消事务后尝试从 QLDB 检索结果时，就会发生此异常。为了缓解此问题，客户端必须在关闭事务之前读取数据。

导出日记账数据

本节列出了当您将分类账中的日记账数据导出到 Amazon S3 存储桶时，QLDB 可能返回的常见异常。有关此功能的更多信息，请参阅从Amazon QLDB 导出日记账数据。

每个异常都包括特定的错误消息，然后是简短的描述和可能的解决方案建议。

AccessDeniedException

消息：用户：*userARN* 无权执行：iam:PassRole on resource: *roleARN*

您无权将 IAM 角色传递给 QLDB 服务。QLDB 要求所有日记账导出请求都有一个角色，并且您必须具有将该角色传递给 QLDB 的权限。该角色为 QLDB 提供了在您指定的 Amazon S3 存储桶中的写入权限。

核实您定义了一个 IAM policy，该策略授予对您指定的 QLDB 服务 (`qldb.amazonaws.com`) 的 IAM 角色资源执行 `PassRole` API 操作的权限。有关策略示例，请参阅Amazon QLDB 基于身份的策略示例。

IllegalArgumentException

消息：QLDB 在验证 S3 配置时遇到错误：*errorCode errorMessage*

导致此错误的一个可能原因是 Amazon S3 中不存在所提供的 Amazon S3 存储桶。或者，QLDB 没有足够的权限将对象写入您指定的 Amazon S3 存储桶。

验证您在导出任务请求中提供的 S3 存储桶名称是否正确。有关命名存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南 中的 存储桶限制和局限。

此外，请确认您是否为指定的存储桶定义了一个向 QLDB 服务 (`qldb.amazonaws.com`) 授予 `PutObject` 和 `PutObjectAcl` 权限的策略。要了解更多信息，请参阅 导出权限。

IllegalArgumentException

消息：在验证 S3 配置时，来自 Amazon S3 的意外响应。来自 S3 的响应：*errorCode errorMessage*

尝试将日记账导出数据写入提供的 S3 失败，并显示提供的 Amazon S3 错误响应。有关可能故障的更多信息，请参阅 Amazon Simple Storage Service 用户指南 中的 Amazon S3 故障排除。

IllegalArgumentException

消息：Amazon S3 存储桶前缀不得超过 128 个字符

日记账导出请求中提供的前缀包含超过 128 个字符。

IllegalArgumentException

消息：开始日期不得早于结束日期

`InclusiveStartTime` 和 `ExclusiveEndTime` 都必须采用 ISO 8601 日期和时间格式以及通用协调时间 (UTC)。

IllegalArgumentException

消息：结束日期不能是未来的日期

`InclusiveStartTime` 和 `ExclusiveEndTime` 都必须采用 `ISO 8601` 日期和时间格式以及 UTC。

IllegalArgumentException

消息：提供的对象加密设置 (S3EncryptionConfiguration) 与 AWS Key Management Service (AWS KMS) 密钥不兼容

您提供了一个 `KMSKeyArn`，并有 `NO_ENCRYPTION` 或 `SSE_S3` 中的一个 `ObjectEncryptionType`。您只能为客户提供对象加密类型 `SSE_KMS` 的托管 AWS KMS key。有关在 Amazon S3 中使用服务器端加密选项的信息，请参阅 Amazon S3 开发人员指南 中的使用服务器端加密保护数据。

LimitExceededException

消息：已超出 2 个同时运行日记账导出任务的限制

QLDB 强制执行两个并发日记账导出任务的默认限制。

流日记账数据

本节列出了当您将分类账中的流日记账数据导出到 Amazon Kinesis Data Streams 时的常见异常。有关此功能的更多信息，请参阅Amazon QLDB 流式传输日记账数据。

每个异常都包括特定的错误消息，然后是简短的描述和可能的解决方案建议。

AccessDeniedException

消息：用户：*`userARN`* 无权执行：iam:PassRole on resource: *`roleARN`*

您无权将 IAM 角色传递给 QLDB 服务。QLDB 要求所有日记账流请求都有一个角色，并且您必须具有将该角色传递给 QLDB 的权限。该角色为 QLDB 提供了在您指定的 Amazon Kinesis Data Streams 资源中的写入权限。

核实您定义了一个 IAM policy，该策略授予对您指定的 QLDB 服务 (`qldb.amazonaws.com`) 的 IAM 角色资源执行 `PassRole` API 操作的权限。有关策略示例，请参阅Amazon QLDB 基于身份的策略示例。

IllegalArgumentException

消息：QLDB 在验证 Kinesis 数据流时遇到错误：来自 Kinesis 的响应：*errorCode errorMessage*

造成此错误的一个可能原因是所提供的 Kinesis 数据流资源不存在。或者，QLDB 没有足够的权限将数据记录写入您指定的 Kinesis 数据流。

验证您在流请求中提供的 Kinesis 数据流是否正确。有关更多信息，请参阅 《Amazon Kinesis Data Streams 开发人员指南》 中的 创建和更新数据流。

此外，请确认您是否为指定的 Kinesis 数据流定义了一个策略，该策略授予 QLDB 服务 (`qldb.amazonaws.com`) 对以下操作的权限。有关更多信息，请参阅流权限。

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

消息：在验证 Kinesis 配置时，来自 Kinesis 数据流的意外响应。来自 Kinesis 的回复：*errorCode errorMessage*

尝试将数据记录写入所提供的 Kinesis 数据流失败，并显示提供的 Kinesis 错误响应。有关更多信息，请参阅 《Amazon Kinesis Data Streams 开发人员指南》 中的 Amazon Kinesis Data Streams 生产者故障排除。

IllegalArgumentException

消息：开始日期不得早于结束日期。

`InclusiveStartTime` 和 `ExclusiveEndTime` 都必须采用 ISO 8601 日期和时间格式以及通用协调时间 (UTC)。

IllegalArgumentException

消息：开始日期不能是未来的日期。

`InclusiveStartTime` 和 `ExclusiveEndTime` 都必须采用 `ISO 8601` 日期和时间格式以及 UTC。

LimitExceededException

消息：已超出 Kinesis 数据流同时运行 5 个日记账流的限制

QLDB 强制执行五个并发日记账流的默认限制。

验证日记账数据

本节列出了在验证分类账中的日记账数据时 QLDB 可能返回的常见异常。有关此功能的更多信息，请参阅Amazon QLDB 中的数据验证。

每个异常包括具体的错误消息，接着是可能引发异常的 API 操作、一个简短的描述以及可能的解决方案建议。

IllegalArgumentException

消息：提供的 Ion 值无效，无法解析。

API 操作：`GetDigest, GetBlock, GetRevision`

在重试请求之前，请务必提供有效的 Amazon Ion 值。

IllegalArgumentException

消息：提供的块地址无效。

API 操作：`GetDigest, GetBlock, GetRevision`

在重试请求之前，请务必提供有效的块地址。地址是一种包含两个字段的 Amazon Ion 结构：即 `strandId` 和 `sequenceNo`。

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

IllegalArgumentException

消息：提供的摘要提示地址的序列号超出了该链路的最新提交记录。

API 操作：`GetDigest, GetBlock, GetRevision`

您提供的摘要提示地址的序号必须小于或等于日记链最新提交的记录的序号。在重试请求之前，请务必提供带有有效序号的摘要提示地址。

IllegalArgumentException

消息：提供的块地址的链路 ID 无效。

API 操作：`GetDigest, GetBlock, GetRevision`

您提供的块地址必须具有与该日记账的链路 ID 相匹配的链路 ID。在重试请求之前，请务必提供一个具有有效链路 ID 的块地址。

IllegalArgumentException

消息：提供的块地址的序列号超出了该链路的最新提交记录。

API 操作：`GetBlock, GetRevision`

您提供的块地址的序号必须小于或等于链路最新提交的记录的序号。在重试请求之前，请务必提供带有有效序号的块地址。

IllegalArgumentException

消息：提供的块地址的链 ID 必须与提供的摘要提示地址的链 ID 相匹配。

API 操作：`GetBlock, GetRevision`

只有当文档修订或块存在于与提供的摘要相同的日记账链中时，才能验证文档修订或块。

IllegalArgumentException

消息：提供的块地址的序列号不得大于所提供的摘要提示地址的序号。

API 操作：`GetBlock, GetRevision`

只有当您提供的摘要包含文档修订或块时，您才能验证它。这意味着它是在摘要提示地址之前提交给日记账的。

IllegalArgumentException

消息：在指定块地址的块中找不到提供的文档 ID。

API 操作：`GetRevision`

您提供的文档 ID 必须存在于您提供的块地址中。在重试请求之前，请确保这两个参数是一致的。

Amazon QLDB PartiQL 参考

Amazon QLDB 支持 PartiQL 查询语言的子集。以下主题介绍 PartiQL 的 QLDB 实现。

> **Note**
>
> - QLDB 不支持所有 PartiQL 操作。
> - QLDB 中的所有 PartiQL 语句都受事务限制的约束，如Amazon QLDB 资源中的限额和限制中所定义。
> - 此参考提供可以使用 QLDB 控制台 或 QLDB Shell 手动运行的 PartiQL 语句的基本语法和用法示例。有关展示如何使用 QLDB 驱动程序以编程方式运行类似语句的代码示例，请参阅驱动程序入门中的教程。

主题

什么是 PartiQL？

PartiQL 在包含结构化数据、半结构化数据和嵌套数据的多个数据存储中提供 SQL 兼容的查询访问。它在 Amazon 中广泛使用，现在可作为许多 AWS 服务 服务(包括 QLDB)的一部分提供。

有关 PartiQL 规范和核心查询语言的教程，请参阅 ParameSQL 文档。

PartiQL 扩展了 SQL-92，以支持 Amazon Ion 数据格式的文档。有关 Amazon Ion 的更多信息，请参阅 Amazon QLDB 中的 Amazon Ion 数据格式参考。

Amazon QLDB 中的 PartiQL

若要 QLDB 中运行 PartiQL 查询，您可以使用以下方法之一：

- QLDB 的AWS Management Console上的PartiQL 编辑器
- 命令行 QLDB Shell
- AWS 提供的 QLDB 驱动程序，用于以编程方式运行查询

有关使用这些方法访问 QLDB 的更多信息，请参阅 访问 Amazon QLDB。

要了解如何控制特定表运行每个 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

关于 QLDB 中 PartiQL 的快速小贴士

以下是在 QLDB 中使用 PartiQL 的提示和最佳实践小贴士：

- 了解并发和事务限制 — SELECT 查询等所有语句都应遵守乐观并发控制 (OCC) 冲突和事务限制，包括 30 秒事务暂停。
- 使用索引 - 使用高基数索引，并运行有针对性的查询来优化语句并避免全表扫描。要了解更多信息，请参阅 优化查询性能。
- 使用相等谓词 - 索引查找需要相等运算符 (= 或 IN)。不等式运算符 (<、>、LIKE、BETWEEN) 不符合索引查找的条件，因此会生成全表扫描。
- 仅使用内部联接 - QLDB 仅支持内部联接。根据最佳实践标准，在为要加入的每个表编制索引的字段上进行联接。为联接条件与相等谓词选择高基数索引。

Amazon QLDB PartiQL 参考惯例

此部分介绍 Amazon QLDB PartiQL 参考 中描述的用于编写 PartiQL 命令、函数和表达式语法的约定。不要将这些约定与 PartiQL 查询语言本身的 语法和语义混淆。

字符	描述
CAPS	采用大写字母的字样为关键字。
[ ]	括号表示可选参数或子句。方括号中的多个参数表示您可选择任意数量的参数。此外，不同行的括号中的参数表示 QLDB 分析程序需要按照参数在语法中列出的顺序获取参数。
|	管道表示您可以在不同参数之间选择。
####	红色斜体字样表示占位符。必须插入适当的值以替换红色斜体字样。
. . .	省略号表示可以重复前面的元素。
'	单引号中的值表示必须输入单引号。在 PartiQL 中，单引号表示字符串值或 Amazon Ion 结构的字段名称。
"	双引号中的值表示您必须输入双引号。在 PartiQL 中，双引号表示带引号标识符。
`	反引号值表示必须输入反引号。在 PartiQL 中，反引号表示 Ion 文本值。

Amazon QLDB 中的数据类型

Amazon QLDB 以 Amazon Ion格式存储文档。Amazon Ion 是一种数据序列化格式（文本形式和二进制编码形式），是 JSON 的超集。下表列出了您可在 QLDB 文档中使用的 Ion 数据类型。

数据类型	描述
`null`	通用的空值
`bool`	布尔值
`int`	任意大小的、有符号整数
`decimal`	任意精度的十进制编码实数
`float`	二进制编码的浮点数 (64 位 IEEE)

数据类型	描述
`timestamp`	任意精度的日期/时间/时区时刻
`string`	Unicode 文本
`symbol`	Unicode 符号原子（标识符）
`blob`	用户定义编码的二进制数据
`clob`	用户定义编码的二进制数据
`struct`	无序名称/值对集合
`list`	有序异构值集合

请参阅 Amazon GitHub 网站上的 Ion 规范文档，了解 Ion core 数据类型的完整列表，以及完整的描述和值格式化细节。

Amazon QLDB 文档

Amazon QLDB 将数据记录存储为文档，这些文档只是插入到表中的Amazon Ion `struct`对象。有关 Ion 规范，请访问 Amazon Ion GitHub 网站。

主题

- Ion 文档结构
- PartiQL-ion 类型映射
- 文档 ID

Ion 文档结构

与 JSON 一样的是，QLDB 文档由以下结构中的名称/值对组成。

```
{
  name1: value1,
  name2: value2,
  name3: value3,
  ...
```

```
  nameN: valueN
}
```

名称为符号标记，其值不受限制。每个名称/值对都称为一个字段。字段的值可以是任何 Ion数据类型，包括以下容器类型：嵌套结构、列表和结构列表。

与 JSON 一样的是，`struct`用大括号 (`{...}`) 表示，`list` 用方括号 (`[...]`) 表示。以下示例是Amazon QLDB 控制台入门中示例数据的文档，其中包含各种类型的值。

```
{
    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**
>
> 在 Ion 中，双引号表示字符串值，未加引号符号表示字段名。但在 PartiQL，单引号表示字符串和字段名。
> 这种语法差异允许 PartiQL 查询语言保持 SQL 兼容性，使 Amazon Ion 数据格式保持 JSON 兼容性。有关 QLDB 中 PartiQL 的语法和语义详细信息，请参阅使用 PartiQL 查询 Ion。

PartiQL-ion 类型映射

在 QLDB，PartiQL 扩展了 SQL 的类型系统，以覆盖 Ion 数据模型。此映射描述如下：

- SQL 标量类型包含在 Ion 对应物之中。例如：
  - `CHAR`和`VARCHAR`是映射到 Ion `string` 类型的 Unicode 序列。
  - `NUMBER`映射至 Ion `decimal`类型。
- Ion 的 `struct`类型等同于 SQL 元组，传统上表示表行。

- 但是，在内容开放且没有架构的情况下，不支持依赖 SQL 元组有序性质的查询 (例如`SELECT *`的输出顺序)。
- 除`NULL`之外，PartiQL 还有一个`MISSING`类型。这是一种专业化`NULL`，表明缺少字段。此类型为必要项，因为 Ion `struct` 字段可能很稀少。

文档 ID

QLDB 会为您插入到表格中的每个文档分配一个文档 ID。所有系统分配的 ID 都是通用唯一标识符 (UUID)，每个标识符都以 Base62 编码的字符串表示 (例如`3Qv67yjXEwB9SjmvkuG6Cp`)。有关更多信息，请参阅Amazon QLDB 中的唯一编号。

每个文档修订版都由文档 ID 和从零开始的版本号组合作为唯一标识。

文档 ID 和版本字段都包含在文档元数据中，您可以在提交视图（系统定义的表格视图）中对其进行查询。有关 QLDB 中的这些视图的更多信息，请参阅核心概念。了解有关元数据的更多信息，请参阅 查询文档元数据。

在 Amazon QLDB 中使用 PartiQL 查询 Ion

当您通过 Amazon QLDB 查询数据时，您以 PartiQL 格式编写语句，但是 QLDB 会以 Amazon Ion 格式返回结果。PartiQL 旨在兼容 SQL，而 Ion 是 JSON 的扩展项。这会导致在查询语句中对数据进行注释的方式与查询结果显示方式在语法上存在差异。

本节介绍使用 QLDB 控制台或 QLDB 控制台 或 QLDB Shell 手动运行 PartiQL 语句的基本语法和语义。

> **Tip**
>
> 当以编程方式运行 PartiQL 查询时，最佳做法是使用参数化语句。可在语句中使用问号 (?) 作为绑定变量占位符，以避免使用这些语法规则。这也更安全和高效。
> 若要了解更多信息，请参阅驱动程序入门中的以下教程：
>
> - Java: 快速入门教程 | 说明书参考
> - .NET: 快速入门教程 | 说明书参考
> - Go: 快速入门教程 | 说明书参考
> - Node.js: 快速入门教程 | 说明书参考
> - Python: 快速入门教程 | 说明书参考

主题

语法和语义

使用 QLDB 控制台或 QLDB Shell 查询 Ion 数据时，以下是 PartiQL 的基本语法和语义：

区分大小写

所有 QLDB 系统对象名称（包括字段名、表名和分类账名称）均区分大小写。

字符串值

在 Ion 中，双引号 ("...") 表示字符串。

在 PartiQL 中，单引号 ('...') 表示字符串。

符号和标识符

在 Ion 中，单引号 ('...') 表示符号。Ion 中称为标识符的符号子集由未加引号的文本表示。

在 PartiQL 中，双引号 ("...") 表示带引号的 PartiQL 标识符，如用作表名的保留字。未加引号的文本表示常规 PartiQL 标识符，例如非保留字的表名。

Ion 文本

在 PartiQL 语句中，任何 Ion 文字都可以用反引号 (`...`) 表示。

字段名称

Ion 字段名称区分大小写。PartiQL 允许您在 DML 语句中通过单引号表示字段名称。这是使用 PartiQL `cast`函数定义符号的简写替代方案。它也比使用反引号表示字面上的 Ion 符号更直观。

文本

PartiQL 查询语言的字面值对应 Ion 数据类型，如下所示：

标量

如果适用，请按照 SQL 语法进行操作，如第 PartiQL-ion 类型映射 节所述。例如：

- `5`
- `'foo'`
- `null`

Structs

也称为多种格式的元组或对象、以及其他数据模型。

用大括号 (`{...}`) 表示，其中`struct`元素用逗号分隔。

- `{ 'id' : 3, 'arr': [1, 2] }`

列表

也被称为数组。

用方括号 (`[...]`) 表示，列表元素以逗号分隔。

- `[ 1, 'foo' ]`

数据包

PartiQL 中的无序集合。

用双尖括号 (`<<...>>`) 表示，数据包元素用逗号隔开。在 QLDB 中，表格可以被视为一个数据包。但是，数据包不能嵌套在表格中的文档中。

- `<< 1, 'foo' >>`

示例

下面是具有各种 Ion 类型`INSERT`语句的语法示例。

```
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`
}
```

反引号表示法

PartiQL 完全涵盖所有 Ion 数据类型，因此您无需使用反引号即可编写任何语句。但是在某些情况下，这种 Ion 文字语法可以让您的语句更清晰、更简洁。

例如，要插入带有 Ion 时间戳和符号值的文档，可仅使用纯粹的 PartiQL 语法编写以下语句。

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

这相当冗长，所以你可改用反引号简化你的语句。

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

您也可以用反引号将整个结构括起，以节省更多的按键次数。

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

> **Important**
>
> 在 PartiQL 中，字符串和符号类别不同。这意味着，即使它们有相同的文本，也不相等。例如，以下 PartiQL 表达式计算为不同 Ion 值。

```
'foo'
```

```
`foo`
```

路径导航

编写数据操作语言 (DML) 或查询语句时，您可以使用路径步骤访问嵌套结构中的字段。PartiQL 支持使用点表示法访问父结构的字段名称。以下示例访问父项 `Vehicle`的 `Model` 字段。

```
Vehicle.Model
```

若要访问列表中的特定元素，可以使用方括号运算符表示从零开始的序数。以下示例访问序数为`SecondaryOwners`的`2`元素。换句话说，这是列表的第三元素。

```
SecondaryOwners[2]
```

别名

QLDB 支持开放内容与架构。因此，当您访问语句中的特定字段时，确保获得预期结果的最佳方法是使用别名。例如，如果您未指定显式别名，则系统会为您的 FROM 源生成一个隐式别名。

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

但是字段名冲突的结果不可预测。如果文档中的嵌套结构中存在另一个名为`VIN`的字段，则此查询返回的`VIN`值可能会让您感到惊讶。作为最佳实践标准，请改写以下语句。此查询声明 `v` 为 `Vehicle` 表的别名。AS 关键字是可选的。

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

当路径进入文档中嵌套集合时，别名特别有用。例如，以下语句声明 `o` 为`VehicleRegistration.Owners`覆盖集合的别名。

```
SELECT o.SecondaryOwners
```

```
FROM VehicleRegistration AS r, @r.Owners AS o
```

此处的 @ 字符在技术上是可选的。但它明确表示你想在里面放置一个Owners结构VehicleRegistration，而不是一个名为不同的集合Owners（如有）。

PartiQL 规范

有关 PartiQL 查询语言的更多信息，请参阅 PartiQL 规范。

Amazon QLDB 中的 PartiQL 命令

PartiQL 扩展了 SQL-92 以支持 Amazon Ion 数据格式的文档。Amazon QLDB 支持以下 PartiQL 命令。

要了解如何控制特定表运行每个 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

> **Note**
>
> - QLDB 不支持所有 PartiQL 命令。
> - QLDB 中的所有 PartiQL 语句都受事务限制的约束，如Amazon QLDB 资源中的限额和限制中所定义。
> - 本参考资料提供了在 QLDB 控制台或 QLDB Shell 上手动运行的 PartiQL 语句的基本语法和用法示例。有关演示如何使用受支持的编程语言运行类似语句的代码示例，请参阅驱动程序入门中的教程。

DDL 语句(数据定义语言)

数据定义语言 (DDL) 是一组用于管理数据库对象（如表和索引）的 PartiQL 语句。您可以使用 DDL 创建和删除这些对象。

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- DROP TABLE

- 取消删除表

DML(数据操作语言)语句

数据操作语言 (DML) 是一组用于管理 QLDB 表中的数据的 PartiQL 语句。可以使用 DML 语句在表中添加、修改或删除数据。

支持以下 DML 和查询语言语句：

- 删除
- FROM（插入、删除或设置）
- INSERT
- SELECT
- 更新

在 Amazon QLDB 中创建索引命令

在 Amazon QLDB 中，使用`CREATE INDEX`命令为表中的文档字段创建索引。

要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

> **Important**
>
> QLDB 需要索引才能高效查找文档。如果没有索引，QLDB 在读取文档时需进行全表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (=或`IN`) 运行带有`WHERE`谓词子句的语句。有关更多信息，请参阅优化查询性能。

创建索引时应注意以下限制：

- 只能在单个顶级字段创建索引。不支持复合索引、嵌套索引、唯一索引以及基于函数的索引。
- 您可以为任何 Ion 数据类型创建索引，其中包括 `list` 和 `struct`。但是，无论 Ion 类型如何，您都只能通过整个 Ion 值进行索引查找。例如，使用 `list` 类型作为索引时，不能按列表中的一个项目进行索引查找。

- 只有使用相等谓词时，查询性能才会得到改善；例如`WHERE indexedField = 123` 或 `WHERE indexedField IN (456, 789)`。

  QLDB 不支持查询谓词不等式。因此，未实施范围过滤扫描。
- 索引字段名称区分大小写，且最大长度可为 128 个字符。
- 在 QLDB 中创建索引具有异步特点。非空表上完成索引所需的时间取决于表的大小。有关更多信息，请参阅管理索引。

主题

语法

```
CREATE INDEX ON table_name (field)
```

参数

table_name

要在其中创建索引的表的名称。表必须已经存在。

表名称区分大小写。

field

要为其创建索引的文档字段的名称。该字段必须为顶级属性。

索引字段名称区分大小写，且最大长度可为 128 个字符。

您可以为任何 Amazon Ion 数据类型创建索引，包括 `list` 和 `struct`。但是，无论 Ion 类型如何，您都只能通过整个 Ion 值进行索引查找。例如，使用 `list` 类型作为索引时，不能按列表中的一个项目进行索引查找。

返回值

`tableId` — 您在其上创建索引的表的唯一 ID。

示例

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

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

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的 CREATE TABLE 命令

在 Amazon QLDB 中使用 `CREATE TABLE`命令创建新表。

表名称很简单，没有命名空间。QLDB 支持开放内容且不强制架构，因此在创建表时不需要定义属性或数据类型。

> **Note**
>
> 要了解如何控制在分类账中运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

主题

- 语法
- 参数

语法

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

参数

table_name

要创建的表的唯一名称。不得存在同名活动表。以下是命名约定：

- 只能包含 1-128 个字母数字字符或下划线字符。
- 必须为首个字样或下划线。
- 其余字符可能是字母数字字符和下划线的任意组合。
- 区分大小写。
- 不能是 QLDB PartiQL 的 保留字。

'***key***': '***value***'

（可选）在创建表资源时要附加至表资源的标签。每个标签都定义为键值对，其中键和值均以单引号表示。每个键值对都在用反引号表示的 Amazon Ion 结构中定义。

目前，只有 *STANDARD* 权限模式分类账支持在创建时对表格进行标记。

返回值

`tableId` - 您创建的表的唯一 ID。

创建标记表

> **Note**
>
> 目前，只有 STANDARD 权限模式分类账支持在创建时对表格进行标记。

或者，您可以通过CREATE TABLE语句中指定标签来标记表资源。有关标签的更多信息，请参阅 为 Amazon QLDB 资源贴标签。下面的示例创建了一个名为 Vehicle 的表，带有标签 environment=production。

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

在创建时为表添加标签，需要同时访问 qldb:PartiQLCreateTable 和 qldb:TagResource 操作。要了解有关 QLDB 资源权限的更多信息，请参阅Amazon MQ 如何与 IAM 协同工作。

通过在创建资源时对其进行标记，无需在创建资源后运行自定义标记脚本。标记表后，您可根据这些标签来控制对表的访问。例如：您只能向具有特定标签的表授予完全访问权限。有关 JSON 策略示例，请参阅基于表格标签对所有操作的完全访问权限。

示例

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的删除命令

在 Amazon QLDB 中，通过创建新的但最终版本的文档，使用DELETE命令将表中的活动文档标记为已删除。此最终修订版表示此文档已被删除。此操作会结束文档生命周期，这意味着无法再创建具有相同文档 ID 的文档修订版。

该操作不可逆。您仍然可以使用历史记录函数查询已删除文档的修订历史记录。

> **Note**
>
> 要了解如何控制对特定表上运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

主题

语法

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

参数

table_name

包含要删除数据的用户表的名称。仅默认用户视图支持 DML 语句。每条语句只能在单个表中运行。

AS ***table_alias***

（可选）用户定义的别名，范围涵盖要从中删除的表。AS 关键字是可选的。

BY ***id_alias***

（可选）用户定义的别名，它绑定到结果集中每个文档的 `id`元数据字段。必须使用BY关键字FROM在子句中声明别名。当您想在查询默认用户视图的同时筛选文档 ID，这很有用。有关更多信息，请参阅通过 BY 子句查询文档 ID。

WHERE ***condition***

待删除文档的选择条件。

> **Note**
>
> 如果省略 WHERE 子句，则表中的所有文档都被删除。

返回值

`documentId` — 您删除的每个文档的唯一 ID。

示例

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的 DROP INDEX 命令

在 Amazon QLDB 中，使用`DROP INDEX`命令删除表上的索引。

> **Note**
>
> 要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

主题

- 语法

语法

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> 该子句 `WITH (purge = true)` 是所有 `DROP INDEX` 语句所必需的，并且 `true` 是目前唯一支持的值。
> 关键字purge区分大小写，并且必须为全小写。

参数

"*indexId*"

待索引的唯一 ID，用双引号表示。

ON *table_name*

要删除索引的表名称。

返回值

`tableId` - 您删除其索引的表的唯一 ID。

示例

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

Amazon QLDB 中的 DROP TABLE 命令

在 Amazon QLDB 中，使用 `DROP TABLE` 命令停用现有表。您可以使用 取消删除表 语句将其重新激活。停用或重新激活表，不会影响其文档或索引。

> **Note**
>
> 要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

主题

语法

```
DROP TABLE table_name
```

参数

table_name

要休眠的表的名称。该表必须已存在且状态为 ACTIVE。

返回值

tableId — 您停用表的唯一 ID。

示例

```
DROP TABLE VehicleRegistration
```

Amazon QLDB 中的 FROM（插入、删除或设置）命令

在 Amazon QLDB 中，以FROM开头的语句是 PartiQL 扩展，允许您在文档中插入和删除特定元素。您也可以使用此语句更新文档中的现有元素，类似于更新命令。

> **Note**
>
> 要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

主题

语法

FROM-INSERT

在现有文档内插入新元素。要在表格中插入新顶级文档，必须使用INSERT。

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

FROM-REMOVE

移除文档中的现有元素，或者移除整个顶级文档。后者在语义上与传统 删除 语法相同。

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

FROM-SET

更新文档中的一项或多项元素。如元素不存在，则将其插入。这在语义上与传统 更新 语法相同。

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
```

```
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

参数

table_name

包含要修改数据的用户表的名称。仅默认用户视图支持 DML 语句。每条语句只能在单个表中运行。

在此子句中，还可包含嵌套在指定表中的一个或多个集合。有关更多信息，请参阅 嵌套集合。

AS ***table_alias***

（可选）用户定义的别名，其范围涵盖要修改的表。SET、REMOVE、INSERT INTO 或 WHERE语句中所用的表格别名必须在FROM语句中声明。AS 关键字是可选的。

BY ***id_alias***

（可选）用户定义的别名，它绑定至结果集中每个文档的 id元数据字段。必须使用 BY 关键字在 FROM 子句中声明别名。当您想在查询默认用户视图的同时筛选文档 ID，这很有用。有关更多信息，请参阅通过 BY 子句查询文档 ID。

WHERE ***condition***

（必需）要修改的文档的选择条件。

> **Note**
>
> 如果省略 WHERE 子句，则表中的所有文档都被修改。

##

待创建或修改的文档元素。

data

元素新值。

AT ***key_name***

在要修改的文档中添加的密钥名称。您必须指定相应的 VALUE 以及秘钥名称。这是在文档中的特定位置插入新 AT 值的必要条件。

嵌套集合

虽然只能对单个表运行 DML 语句，但可以将此表中文档中的嵌套集合指定为其他来源。您为嵌套集合声明的每个别名都可以在 WHERE子句和SET、INSERT INTO或REMOVE子句中使用。

例如，以下语句的FROM 来源包括 VehicleRegistration 表和嵌套 Owners.SecondaryOwners 结构。

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

此示例更新了特定SecondaryOwners列表元素，其中包含'1N4AL11D75C109151'中的VIN个VehicleRegistration文档中的'abc123' 的PersonId。此表达式允许您按其值而非其索引指定列表中的元素。

返回值

documentId — 您更新或删除的每个文档的唯一 ID。

示例

修改文档中的一项元素。如果此元素不存在，则将其插入。

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

修改或插入元素并筛选系统分配的文档id元数据字段。

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

修改文档中Owners.SecondaryOwners列表中第一个元素的PersonId字段。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

移除文档中的现有元素。

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

从表格中删除整个文档。

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

移除VehicleRegistration表格中文档中Owners.SecondaryOwners列表的第一个元素。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

{'Mileage':26500}以顶级名称/值对的形式插入到表格中的文档中。Vehicle

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

{'PersonId':'abc123'}以名称/值对的形式附加至VehicleRegistration表格中Owners.SecondaryOwners文档的字段中。请注意，语句Owners.SecondaryOwners必须已经存在且必须是列表数据类型才会生效。否则，INSERT INTO子句中必须使用关键字AT。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

将{'PersonId':'abc123'}以文档现有Owners.SecondaryOwners列表中的首个元素插入。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

将多个名称/值对附加到文档中的现有Owners.SecondaryOwners列表中。

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的 INSERT 命令

在 Amazon QLDB 中，使用`INSERT`命令将一个或多个 Amazon Ion 文档添加到表格中。

> **Note**
>
> 要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅。请参阅《Amazon QLDB 开发人员》中的标准权限模式入门

主题

语法

插入单个文档。

```
INSERT INTO table_name VALUE document
```

插入多个文档。

```
INSERT INTO table_name << document, document, ... >>
```

参数

table_name

要在其中插入数据的用户表的名称。表必须已经存在。仅默认用户视图支持 DML 语句。

##

有效的 QLDB 文档。您必须指定至少一个文档。必须用逗号分隔多个文档。

文档必须用大括号 (`{...}`) 表示。

文档中的每个字段名称都是一个区分大小写的 Ion 符号，在 PartiQL 中可以用单引号 (`'...'`) 表示。

字符串值也用单引号 (`'...'`) 在 PartiQL 中表示。

任何 Ion 文字都可以用反引号 (`` `...` ``) 表示。

> **Note**
>
> 双尖括号 (`<<...>>`) 表示无序集合(在 PartiQL 中称为数据包)，只有在要插入多个文档时才需要双尖括号。

返回值

`documentId`— 您插入的每个文档的唯一 ID。

示例

插入单个文档。

```
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`
}
```

此语句返回您插入的每个文档的唯一 ID，如下所示。

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

插入多个文档。

```
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'
}
>>
```

此语句返回您插入的文档的唯一 ID，如下所示。

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的 SELECT 命令

在 Amazon QLDB 中，使用SELECT命令从一个或多个表中检索数据。QLDB 中的每个 SELECT 查询都是在事务中处理的，并且受事务超时限制的约束。

结果的顺序不指定，可能因每个SELECT查询而异。在 QLDB 中，任何查询都不应该依赖结果顺序。

要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅。请参阅《Amazon QLDB 开发人员》中的标准权限模式入门

> **Warning**
>
> 当你在没有索引查找的情况下运行查询时，它会调用全表扫描。PartiQL 之所以支持此类查询，是因为其与 SQL 兼容。但是，切勿在 QLDB 中对生产用例运行表扫描。表扫描可能会导致大型表出现性能问题，包括并发冲突与事务超时。
>
> 为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (`WHERE indexedField = 123`或`WHERE indexedField IN (456, 789)`) 运行带有`WHERE`谓词子句的语句。有关更多信息，请参阅优化查询性能。

主题

- 语法
- 参数
- Joins
- 嵌套查询限制
- 示例
- 使用驱动程序以编程方式运行

语法

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
[ WHERE condition ]
```

参数

VALUE

表达式的限定符，它使查询返回原始数据类型值，而非将值封装在元组结构中。

expression

(必需)从 * 通配符形成的投影，或者结果集的一个或多个属性名称或文档路径的投影列表。表达式可以包括对 PartiQL 函数 或通过 PartiQL 运算符 修改的字段的调用。

AS *field_alias*

（可选）在最终结果集中使用字段的临时别名。AS 关键字是可选的。

如果您没有为不是简单列名的表达式指定别名，则结果集将对该字段应用默认名称。

从 *#*

要查询的来源。当前支持的唯一来源是表名、表之间的内部联接、嵌套SELECT查询 (以嵌套查询限制) 和表的历史函数 调用为准。

您必须指定至少一个源。必须用逗号分隔多个源。

AS *source_alias*

（可选）用户定义的别名，范围涵盖要更新的源。SELECT、WHERE 语句中所用的源别名必须在 FROM 语句中声明。AS 关键字是可选的。

AT *idx_alias*

（可选）用户定义的别名，它绑定至源列表中每个元素的索引(序数)。必须使用 AT 关键字在 FROM 子句中声明别名。

BY *id_alias*

（可选）用户定义的别名，它绑定至结果集中每个文档的 id元数据字段。必须使用 BY 关键字在 FROM 子句中声明别名。当您想在查询默认用户视图的同时投影或筛选文档 ID，这很有用。有关更多信息，请参阅通过 BY 子句查询文档 ID。

WHERE *condition*

查询的选择标准和联接标准（如适用）。

> **Note**
>
> 如果省略 WHERE 子句，则检索表中的所有文档。

Joins

目前仅支持内部联接。您可以使用显式 INNER JOIN 子句编写内部联接查询，如下所示。在此语法中，JOIN必须与ON配对，并且INNER 关键字是可选的。

```
SELECT expression
```

```
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

或者，您可以使用隐式语法编写内部联接，如下所示。

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

嵌套查询限制

可以在 SELECT 表达式和 FROM 源代码中编写嵌套查询（子查询）。主要限制是，只有最外层的查询才能访问全局数据库环境。例如，假设您有一个包含表VehicleRegistration和Person的分类账。以下嵌套查询无效，因为内部SELECT试图访问Person。

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

而以下嵌套查询是有效的。

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

示例

以下查询显示了一个基本的 SELECT，其中包含使用IN运算符的标准WHERE谓词子句。

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

下图显示了使用字符串筛选条件的 SELECT 光投影。

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

下面显示了用于扁平化嵌套数据的相关子查询。此处的 @ 字符在技术上是可选的。但它明确表示你想在里面放置一个Owners结构VehicleRegistration，而不是一个名为不同的集合Owners（如有）。有关更多背景信息，请参阅 处理数据和历史记录 一章中的嵌套数据。

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

下图显示了SELECT 列表中投射嵌套数据的子查询和隐式内部联接。

```
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')
```

下面显示了一个显式的内部联接。

```
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')
```

下图显示了使用 BY 子句对文档 id 元数据字段的投影。

```
SELECT
    r_id,
    r.VIN
FROM
```

```
    VehicleRegistration AS r BY r_id
WHERE
    r_id = 'documentId'
```

以下内容分别使用BY来联接DriversLicense和Person表的id字段PersonId和文档字段。

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

以下内容使用子已提交视图句分别连接DriversLicense和Person表的id字段PersonId和文档字段。

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

以下内容返回表 VehicleRegistration 中文 档Owners.SecondaryOwners 列表中每个人的PersonId和索引（序数）。

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的更新命令

在 Amazon QLDB 中，使用UPDATE命令来修改文档中一个或多个元素的值。如元素不存在，则将其插入。

您也可以使用此命令在文档中显式插入和删除特定元素，类似于FROM（插入、删除或设置）语句。

> **Note**
>
> 要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅。请参阅《Amazon QLDB 开发人员》中的标准权限模式入门

主题

语法

UPDATE-SET

更新文档中的一项或多项元素。如元素不存在，则将其插入。这在语义上与 FROM-SET 语句相同。

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

UPDATE-INSERT

在现有文档内插入新元素。要在表格中插入新顶级文档，必须使用INSERT。

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

UPDATE-REMOVE

移除文档中的现有元素，或者移除整个顶级文档。后者在语义上与传统 删除 语法相同。

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
```

```
[ WHERE condition ]
```

参数

table_name

包含要修改数据的用户表的名称。仅默认用户视图支持 DML 语句。每条语句只能在单个表中运行。

AS ***table_alias***

（可选）用户定义的别名，其范围涵盖要修改的表。AS 关键字是可选的。

BY ***id_alias***

（可选）用户定义的别名，它绑定至结果集中每个文档的 `id`元数据字段。必须使用 BY 关键字在 UPDATE 子句中声明别名。当您想在查询默认用户视图的同时筛选文档 ID，这很有用。有关更多信息，请参阅通过 BY 子句查询文档 ID。

##

待创建或修改的文档元素。

data

元素新值。

AT ***key_name***

在要修改的文档中添加的密钥名称。您必须指定相应的 VALUE 以及秘钥名称。这是在文档中的特定位置插入新 AT 值的必要条件。

WHERE ***condition***

（必需）要修改的文档的选择条件。

> **Note**
>
> 如果省略 WHERE 子句，则表中的所有文档都被修改。

返回值

`documentId` — 您更新的每个文档的唯一 ID。

示例

更新文档中的表。如果此字段不存在，则将其插入。

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

在系统分配的文档id元数据字段上进行筛选。

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

覆盖整个文档。

```
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'
```

修改文档中Owners.SecondaryOwners列表中第一个元素的PersonId字段。

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
WHERE r.VIN = '1N4AL11D75C109151'
```

{'Mileage':26500}以顶级名称/值对的形式插入到表格中的文档中。Vehicle

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

{'PersonId':'abc123'}以名称/值对的形式附加至VehicleRegistration表格中Owners.SecondaryOwners文档的字段中。请注意，语句Owners.SecondaryOwners必须已经存在且必须是列表数据类型才会生效。否则，INSERT INTO子句中必须使用关键字AT。

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

将{'PersonId':'abc123'}以文档现有Owners.SecondaryOwners列表中的首个元素插入。

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

将多个名称/值对附加到文档中的现有Owners.SecondaryOwners列表中。

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

移除文档中的现有元素。

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

从表格中删除整个文档。

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

移除VehicleRegistration表格中文档中Owners.SecondaryOwners列表的第一个元素。

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

使用驱动程序以编程方式运行

要了解如何使用 QLDB 驱动程序以编程方式运行此语句，请参阅驱动程序入门中的以下教程：

- Java: 快速入门教程 | 说明书参考
- .NET: 快速入门教程 | 说明书参考
- Go: 快速入门教程 | 说明书参考
- Node.js: 快速入门教程 | 说明书参考
- Python: 快速入门教程 | 说明书参考

Amazon QLDB 中的 UNDROP TABLE 命令

在 Amazon QLDB 中，使用 `UNDROP TABLE` 命令重新激活之前删除的表（即已停用）。停用或重新激活表，不会影响其文档或索引。

> **Note**
>
> 要了解如何控制对特定表运行此 PartiQL 命令的访问权限，请参阅。请参阅《Amazon QLDB 开发人员》中的标准权限模式入门

主题

- 语法
- 参数
- 返回值
- 示例

语法

```
UNDROP TABLE "tableId"
```

参数

"*tableId*"

待索引的表的唯一 ID，用双引号表示。

该表必须先前已被删除，这意味着它存在于系统目录表，information_schema.user_tables并且状态为INACTIVE。也不能存在同名活动现有表。

返回值

tableId - 您重新激活的表的唯一 ID。

示例

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Amazon QLDB 中的 PartiQL 函数

Amazon QLDB 中的 PartiQL 支持以下 SQL 标准函数的内置版本。

> **Note**
>
> QLDB 当前不支持任何未包含在此列表中的 SQL 函数。
> 本函数参考基于 PartiQL 文档内置函数。

未知类型（空且缺失）传播

除非另有说明，否则所有函数都会传播空值和缺失参数值。如果任何函数参数为NULL 或 MISSING，则NULL 或 MISSING的传播定义为返回NULL。以下是这种传播示例。

```
CHAR_LENGTH(null)    -- null
CHAR_LENGTH(missing) -- null (also returns null)
```

聚合函数

- AVG
- COUNT
- MAX
- MIN
- SIZE

- SUM

条件函数

- COALESCE
- EXISTS
- NULLIF

日期和时间函数

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

标量函数

- TXID

字符串函数

- CHAR_LENGTH
- CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

数据类型格式设置函数

- CAST
- TO_STRING

- TO_TIMESTAMP

Amazon QLDB 中的 AVG 函数

在 Amazon QLDB 中，使用 AVG 函数返回输入表达式值的平均值（算术平均值）。此函数处理数值，而忽略空值或缺失值。

语法

```
AVG ( expression )
```

Arguments

expression

对其执行函数的数字数据类型字段名称或表达式。

数据类型

支持的参数类型：

- `int`
- `decimal`
- `float`

返回类型: `decimal`

示例

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

相关函数

- COUNT
- MAX
- MIN

- SIZE
- SUM

Amazon QLDB 中的 CAST 功能

在 Amazon QLDB 中，使用 CAST 函数将给定表达式计算为一个值，并将该值转换为指定目标数据类型。如果无法进行转换，则该函数将返回错误。

语法

```
CAST ( expression AS type )
```

Arguments

expression

评估函数转换值的字段名称或表达式。转换 null 值将返回 null。此参数可以是任何支持数据类型的参数。

type

要转换的目标数据类型名称。此参数可以是任何支持数据类型的参数之一。

返回类型

type 参数指定的数据类型。

示例

以下示例显示了未知类型（NULL或MISSING）传播。

```
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)
```

任何非未知类型的值都不能转换为 NULL或MISSING。

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

以下示例显示了转换 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'``)
```

以下示例显示了转换 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
```

以下示例显示了转换 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
```

以下示例显示了转换 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
```

以下示例显示了转换 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
```

以下示例显示了转换 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'
```

以下示例显示了转换 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"
```

以下示例显示了转换 AS struct。

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

以下示例显示了转换 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
```

以下示例为可运行语句，其中包括前面的一些示例。

```
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"
```

相关函数

- TO_STRING
- TO_TIMESTAMP

Amazon QLDB 中的 CHAR_LENGTH 函数

在 Amazon QLDB 中，使用CHAR_LENGTH函数返回指定字符串中的字符数，其中字符 定义为单个 unicode 代码点。

语法

```
CHAR_LENGTH ( string )
```

CHAR_LENGTH是Amazon QLDB 中的 CHARACTER_LENGTH 函数的同义词。

Arguments

string

函数计算的数据类型的 string 字段名称或表达式。

返回类型

int

示例

```
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)
```

相关函数

- LOWER
- SUBSTRING
- TRIM
- UPPER

Amazon QLDB 中的 CHARACTER_LENGTH 函数

CHAR_LENGTH 函数的同义词。

请参阅Amazon QLDB 中的 CHAR_LENGTH 函数。

Amazon QLDB 中的 COALESCE 函数

在 Amazon QLDB 中，给定一个或多个参数列表，使用COALESCE函数按从左到右的顺序计算参数，并返回第一个不是未知类型值（NULL 或 MISSING）。如果所有参数类型均未知，则结果为NULL。

该 COALESCE 函数不会传 播NULL和MISSING。

语法

```
COALESCE ( expression [, expression, ... ] )
```

Arguments

expression

函数计算的、一个或多个字段名称或表达式的列表。每个参数都可以是任何支持的 数据类型。

返回类型

任何支持数据类型。返回类型为NULL，或与计算非空值和非缺失值的第一个表达式的类型相同。

示例

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>   -- "string"
```

相关函数

- EXISTS
- NULLIF

Amazon QLDB 中的 COUNT 函数

在 Amazon QLDB 中，使用函数COUNT 返回由给定表达式定义的文档数量。此函数具有两种变体：

- `COUNT(*)` — 计算目标表中的所有文档，无论它们是否包含空值或缺失值。
- `COUNT(expression)` — 计算特定的现有字段或表达式中具有非空值的文档数。

> **Warning**
>
> COUNT 函数未经过优化，因此我们不建议在没有索引查询的情况下使用它。当您在没有索引查找的情况下运行查询时，它会调用全表扫描。这可能会导致大型表出现性能问题，包括并发冲突与事务超时。

为避免表扫描，必须在索引字段或文档 ID 上使用相等运算符 (=或IN) 运行带有WHERE谓词子句的语句。有关更多信息，请参阅优化查询性能。

语法

```
COUNT ( * | expression )
```

Arguments

expression

对其执行函数的字段名称或表达式。此参数可以是任何支持数据类型的参数。

返回类型

`int`

示例

```
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
```

相关函数

- AVG
- MAX
- MIN
- SIZE
- SUM

Amazon QLDB 中的 DATE_ADD 函数

在 Amazon QLDB 中，使用DATE_ADD函数将给定的时间戳值按指定的间隔递增。

语法

```
DATE_ADD( datetimepart, interval, timestamp )
```

Arguments

datetimepart

对其执行函数的日期或时间段。该参数可以是下列项之一：

- year
- month
- day
- hour
- minute
- second

interval

指定要添加到给定*###*的间隔的整数。负整数减去时间间隔。

timestamp

函数递增数据类型的timestamp字段名称或表达式。

Ion 时间戳文本值可以用反引号 (`...`) 表示。有关时间戳值的格式详细信息和示例，请参阅 Amazon Ion 规格文档中的时间戳。

返回类型

timestamp

示例

```
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
```

相关函数

- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Amazon QLDB 中的 DATE_DIFF 函数

在 Amazon QLDB 中，使用 DATE_DIFF 函数返回两个给定时间戳中指定日期部分之间的差值。

语法

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

Arguments

datetimepart

对其执行函数的日期或时间段。该参数可以是下列项之一：

- year
- month
- day
- hour
- minute

- second

timestamp1、*timestamp2*

函数比较的数据类型 timestamp 的两个字段名称或表达式。如果 *timestamp2* 晚于*timestamp1*，则结果为正。如果 *timestamp2* 早于 *timestamp1*，则结果为负。

Ion 时间戳文本值可以用反引号 (`` `...` ``) 表示。有关时间戳值的格式详细信息和示例，请参阅 Amazon Ion 规格文档中的时间戳。

返回类型

int

示例

```
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
```

相关函数

- DATE_ADD
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Amazon QLDB 中存在函数

在 Amazon QLDB 中，指定一个 PartiQL 值，如果该值是非空集合则使用EXISTS函数返回TRUE。否则，此函数返回 FALSE。如果输入 EXISTS 不是容器，则结果为FALSE。

该 EXISTS 函数不会传 播NULL和MISSING。

语法

```
EXISTS ( value )
```

Arguments

#

函数评估的字段名称或表达式。此参数可以是任何支持数据类型的参数。

返回类型

bool

示例

```
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
```

相关函数

- COALESCE
- NULLIF

Amazon QLDB 中的 EXTRACT 函数

在 Amazon QLDB 中，使用 EXTRACT 函数返回给定时间戳中指定日期或时间段的整数值。

语法

```
EXTRACT ( datetimepart FROM timestamp )
```

Arguments

datetimepart

函数提取的日期或时间段。该参数可以是下列项之一：

- year
- month
- day
- hour
- minute
- second
- timezone_hour
- timezone_minute

timestamp

函数从中提取的数据类型 timestamp 的字段名称或表达式。如果此参数是未知类型 (NULL或MISSING)，则函数返回NULL。

Ion 时间戳文本值可以用反引号 (`...`) 表示。有关时间戳值的格式详细信息和示例，请参阅 Amazon Ion 规格文档中的时间戳。

返回类型

`int`

示例

```
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
```

相关函数

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Amazon QLDB 中的 LOWER 函数

在 Amazon QLDB 中，使用函数 `LOWER` 将给定字符串中的所有大写字符转换为小写字符。

语法

```
LOWER ( string )
```

Arguments

`string`

函数转换的数据类型 `string` 的字段名称或表达式。

返回类型

`string`

示例

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

相关函数

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

Amazon QLDB 中的 MAX 函数

在 Amazon QLDB 中，使用MAX函数返回一组数值中的最大值。

语法

```
MAX ( expression )
```

Arguments

expression

对其执行函数的数字数据类型字段名称或表达式。

数据类型

支持的参数类型：

- `int`
- `decimal`
- `float`

支持的返回类型：

- `int`
- `decimal`
- `float`

示例

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

相关函数

- AVG
- COUNT
- MIN
- SIZE
- SUM

Amazon QLDB 中的 MIN 函数

在 Amazon QLDB 中，使用`MIN` 函数返回一组数值中的最小值。

语法

```
MIN ( expression )
```

Arguments

`expression`

对其执行函数的数字数据类型字段名称或表达式。

数据类型

支持的参数类型：

- int
- decimal
- float

支持的返回类型：

- int
- decimal
- float

示例

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

相关函数

- AVG
- COUNT
- MAX
- SIZE
- SUM

Amazon QLDB 中的 NULLIF 函数

在 Amazon QLDB 中，指定两个表达式，如果两个表达式的计算结果为相同值，则使用NULLIF 函数返回 NULL。否则该函数会返回空，对其执行第一个表达式计算结果。

该 NULLIF 函数不会传播 NULL和MISSING。

语法

```
NULLIF ( expression1, expression2 )
```

Arguments

expression1、*expression2*

函数对比的两个字段名称或表达式。这些参数可以是任何支持数据类型的参数。

返回类型

任何支持数据类型。返回类型为 NULL 或与第一个表达式的类型相同。

示例

```
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
```

相关函数

- COALESCE
- EXISTS

Amazon QLDB 中的 SIZE 函数

在 Amazon QLDB 中，使用 SIZE 函数返回给定容器数据类型（列表、结构或袋子）中元素的数量。

语法

```
SIZE ( container )
```

Arguments

##

函数运行其上的容器字段名称或表达式。

数据类型

支持的参数类型：

- 列表
- 结构
- 包

返回类型: `int`

如果的输入 `SIZE` 不是容器，则该函数会引发错误。

示例

```
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
```

相关函数

- AVG
- COUNT
- MAX
- MIN

- SUM

Amazon QLDB 中的 SUBSTRING 函数

在 Amazon QLDB 中，使用 SUBSTRING 函数从给定字符串返回子字符串。子字符串从指定起始索引开始，以字符串的最后一个字符或指定长度结束。

语法

```
SUBSTRING ( string, start-index [, length ] )
```

Arguments

string

要从中提取子字符串的数据类型string的字段名称或表达式。

start-index

开始提取的起始位置。此数字可以为负。

的首字符索引为 1。

length

（可选）要从*###*中提取的字符(代码点)数，从*start-index*开始，结束于 (*start-index* + *length*) - 1。换句话说，就是子字符串的长度。此数字不能为负。

如果未提供此参数，则函数将一直运行至*###*末尾。

返回类型

string

示例

```
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"
```

相关函数

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

Amazon QLDB 中的 SUM 函数

在 Amazon QLDB 中，使用 SUM 函数返回输入字段值或表达式值的和。此函数处理数值，而忽略空值或缺失值。

语法

```
SUM ( expression )
```

Arguments

expression

对其执行函数的数字数据类型字段名称或表达式。

数据类型

支持的参数类型：

- `int`
- `decimal`

- `float`

支持的返回类型：

- `int` — 对于整数参数
- `decimal` — 对于十进制或浮点数参数

示例

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

相关函数

- AVG
- COUNT
- MAX
- MIN
- SIZE

Amazon QLDB 中的 TO_STRING 函数

在 Amazon QLDB 中，使用 `TO_STRING` 函数以指定格式模式返回给定时间戳字符串表示形式。

语法

```
TO_STRING ( timestamp, 'format' )
```

Arguments

`timestamp`

函数转换为字符串的数据类型 `timestamp` 的字段名称或表达式。

Ion 时间戳文本值可以用反引号 (`` `...` ``) 表示。有关时间戳值的格式详细信息和示例，请参阅 Amazon Ion 规格文档中的时间戳。

format

指定结果格式模式的字符串文本，按其日期部分表示格式。有关有效格式，请参阅时间戳格式字符串。

返回类型

`string`

示例

```
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"
```

相关函数

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_TIMESTAMP
- UTCNOW

Amazon QLDB 中的 TO_TIMESTAMP 函数

在 Amazon QLDB，如果给定一个表示时间戳的字符串，则使用函数TO_TIMESTAMP将该字符串转换为timestamp数据类型。这是 TO_STRING 的逆运算。

语法

```
TO_TIMESTAMP ( string [, 'format' ] )
```

Arguments

string

函数转换为时间戳的数据类型 string 的字段名称或表达式。

format

（可选）一个字符串文本，用于定义其日期部分中的输入###格式。有关有效格式，请参阅时间戳格式字符串。

如果省略此参数，则该函数假定###采用标准 Ion 时间戳格式。建议使用此函数解析 Ion 时间戳。

使用单字符格式符号 (例如y、M、d、H、h、m、s) 时，可选择零填充，但是零填充变体为必填项（如yyyy、MM、dd、HH、 hh、mm、ss）。

对两位数年份 (格式符号yy) 给予特殊处理。大于或等于 70 的值加上 1900，小于 70 的值加上 2000。

月份名称和上午或下午指标不区分大小写。

返回类型

timestamp

示例

```
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
```

相关函数

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- UTCNOW

Amazon QLDB 中的 TRIM 功能

在 Amazon QLDB 中，使用TRIM函数通过删除前导和尾随的空格或一组指定的字符来修剪给定字符串。

语法

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

Arguments

LEADING

（可选）表示要从###开头删除空格或指定字符。如果未指定，则默认值为 BOTH。

TRAILING

（可选）表示要从###末尾删除空格或指定字符。如果未指定，则默认值为 BOTH。

BOTH

（可选）表示要从###的开头和结尾同时删除前导和结尾的空格或指定字符。

characters

（可选）要删除的字符集，指定为string。

如未提供此参数，则会删除空格。

string

函数修剪的数据类型 string 的字段名称或表达式。

返回类型

string

示例

```
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         "
```

相关函数

- CHAR_LENGTH
- LOWER
- SUBSTRING
- UPPER

Amazon QLDB 中的 TXID 函数

在 Amazon QLDB 中，使用TXID函数返回您正在运行的当前语句的唯一事务 ID。这是将当前事务提交至日志账时分配给文档txId元数据字段的值。

语法

```
TXID()
```

Arguments

无

返回类型

`string`

示例

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

Amazon QLDB 中的 UPPER 函数

在 Amazon QLDB 中，使用UPPER函数将给定字符串中的所有小写字符转换为大写字符。

语法

```
UPPER ( string )
```

Arguments

`string`

函数转换的数据类型的 `string` 字段名称或表达式。

返回类型

`string`

示例

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

相关函数

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

Amazon QLDB 中的 UTCNOW 函数

在 Amazon QLDB 中，使用UTCNOW函数将以协调世界时 (UTC) 表示的当前时间返回为timestamp。

语法

```
UTCNOW()
```

此函数要求您在 SELECT查询中指定FROM子句。

Arguments

无

返回类型

timestamp

示例

```
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'
```

相关函数

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING

- TO_TIMESTAMP

时间戳格式字符串

本节提供时间戳格式字符串参考信息。

时间戳格式字符串适用于 TO_STRING 和 TO_TIMESTAMP 函数。这些字符串可以包含日期部分分隔符(例如 -、/ 或 :) 和以下格式符号。

格式	示例	描述
yy	70	两位数字的年份
y	1970	四位数字的年份
yyyy	1970	以零填充的四位数字的年份
M	1	月份整数
MM	01	以零填充的一年中的月份整数
MMM	Jan	月份缩写
MMMM	一月	月份全名
d	2	一个月中的日 (1-31)
dd	02	以零填充的一个月中的日 (01-31)
a	AM 或 PM	经线标识符(适用于 12 小时制)
h	3	小时(24 小时制，01–12)
hh	03	未填充小时 (12 小时制，01—12)
H	3	小时(24 小时制，00–23)
HH	03	空白小时 (24 小时制，00—23)

格式	示例	描述
m	4	分钟数 (00—59)
mm	04	零填充分钟 (00—59)
s	5	秒数 (00—59)
ss	05	零填充秒 (00—59)
S	0	一秒钟的若干分之几(精度：0.1，范围：0.0-0.9)
SS	06	一秒钟的若干分之几(精度：0.01，范围：0.0-0.99)
SSS	060	一秒钟的若干分之几(精度：0.001，范围：0.0-0.999)
X	+07 或 Z	UTC 小时数偏移，如果偏移为 0，则为“Z”
XX	+0700 或 Z	UTC 小时和分钟数偏移，如果偏移为 0，则为“Z”
XXX	+ 07:00 或 Z	UTC 小时和分钟数偏移，如果偏移为 0，则为“Z”
x	+07	与 UTC 的偏移(以小时为单位)
xx	+0700	UTC 小时和分钟数偏移
xxx	-8:00	UTC 小时和分钟数偏移

Amazon QLDB 中的 Partiql 存储进程

在 Amazon QLDB 中，您可使用EXEC命令按照以下语法运行 PartiQL 存储过程。

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB 仅支持以下系统存储过程：

主题

Amazon QLDB 中的 REDACT_REVISION 存储进程

> **Note**
>
> 2021 年 7 月 22 日之前创建的任何分类账目前都不符合编校资格。您可在 Amazon QLDB 控制台查看分类账的创建时间。

在 Amazon QLDB 中，使用 `REDACT_REVISION` 存储过程永久删除索引存储和日记账存储中的单个非活动文档修订版。此存储过程将删除指定版本的所有用户数据。但是，它将使日记账序列和文档元数据(包括文档 ID 和哈希) 保持不变。该操作不可逆。

指定的文档修订版本必须是在历史记录中处于非活动状态的修订版。文档的最新有效修订版不符合此编校条件。

通过运行此存储过程提交编校请求后，QLDB 将异步处理数据的编校。修订完成后，指定修订的用户数据(由 `data` 结构表示)将被新的 `dataHash` 字段替换。该字段的值是已移除 `data` 结构的 Amazon Ion 哈希。因此，分类账保持了其整体数据的完整性，并通过现有验证 API 操作保持加密可验证性。

有关使用示例数据编校操作的示例，请参见对文档修订版执行编校中的编校示例。

> **Note**
>
> 要了解如何控制在指定表中运行此 PartiQL 命令的访问权限，请参阅请参阅《Amazon QLDB 开发人员》中的标准权限模式入门。

主题

- 示例

修订注意事项和限制

开始在 Amazon QLDB 中进行数据编校之前，请务必查看以下注意事项和限制：

- REDACT_REVISION 存储过程以单个非活动文档修订版中的用户数据为目标。若要编校多个修订版，必须为每个修订版运行一次存储进程。您可为每笔事务编校一个修订版。
- 若要编校文档修订版中的特定字段，必须先使用单独的数据操作语言 (DML) 语句修改修订版。有关更多信息，请参阅编校修订版本中的特定字段。
- QLDB 收到编校请求后，您将无法取消或更改该请求。若要确认编校是否已完成，您可以检查修订的`data`结构是否已被`dataHash`字段所取代。要了解更多信息，请参阅 检查编校是否已完成。
- 编校对 QLDB 服务之外复制的任何 QLDB 数据没有影响。这包括 Amazon S3 的任何导出以及 Amazon Kinesis Data Streams 的任何导出。您必须使用其他数据保留方法管理存储在 QLDB 之外的任何数据。
- 编校对日记账中记录的 PartiQL 语句的文字值没有影响。根据最佳实践标准，您应使用变量占位符而非文字值，以编程方式运行参数化语句。占位符以问号 (?) 的形式写在日记账内，而不是任何可能需要编校的敏感信息。

  若要了解如何使用 QLDB 驱动程序以编程方式运行 PartiQL 语句，请参阅驱动程序入门中每种支持的编程语言的教程。

语法

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Arguments

`` `block-address` ``

要编校的文档修订版本的日记账数据块位置。地址是一种包含两个字段的 Amazon Ion 结构：`strandId` 和 `sequenceNo`。

这是一个 Ion 字面值，以反引号表示。例如：

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

要了解如何查找数据块地址，请参阅 查询文档元数据。

'*table-id*'

待编校修订版本的表的唯一 ID，用单引号表示。

若要了解如何查找表 ID，请参阅 查询系统目录。

'*document-id*'

待编校修订版本的唯一文档 ID，用单引号表示。

若要了解如何查找文档 ID，请参阅 查询文档元数据。

返回值

表示要编校文档修订版的 Amazon Ion 结构，格式如下。

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

返回结构字段

- `blockAddress` — 要编校的修订版本的日记账数据块位置。地址包含以下两个字段。
  - `strandId` — 包含数据块的日记账链的唯一 ID。
  - `sequenceNo` — 一个索引号，用于指定数据块在链中的位置。
- `tableId` — 您正在编校其修订版本的表的唯一 ID。
- `documentId` — 要编校的修订版本的唯一文档 ID。
- `version` — 要编校的文档修订版的版本号。

下面是含采样数据的返回结构示例。

```
{
```

```
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

示例

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

Amazon QLDB 中的 PartiQL 运算符

Amazon QLDB 中的 PartiQL 支持以下 SQL 标准运算符。

> **Note**
>
> DynamoDB 当前不支持任何未包含在此列表中的 SQL 运算符。

算术运算符

操作符	描述
+	Add
-	Subtract
*	Multiply
/	Divide
%	取模

比较运算符

操作符	描述
=	等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
<>	不等于

逻辑运算符

操作符	描述
AND	如果 AND 分隔的所有条件都为 TRUE，则为 TRUE
BETWEEN	如果运算数在比较范围内，则为 TRUE
IN	如果运算数等于表达式列表中的一个，则为 TRUE
IS	TRUE如果运算数是给定的数据类型，包括 NULL 和 MISSING
LIKE	如果操作数与模式匹配，则为 TRUE
NOT	反转给定布尔表达式的值
OR	TRUE如果 OR 分隔的任意条件为 TRUE

字符串运算符

操作符	描述
||	联接位于 || 符号的任意一侧的两个字符串并返回联接后的字符串。如果一个或两个表达式都为 NULL，则联接的结果为空。

Amazon QLDB 中的保留关键词

下面是 Amazon QLDB 中的 PartiQL 保留关键字的列表。您可以使用保留关键字作为带双引号的标识符 (例如`"user"`)。有关 QLDB 中 PartiQL 引用惯例的信息，请参阅使用 PartiQL 查询 Ion。

> **Important**
>
> 此列表中的关键字都被视为保留关键词，因为 PartiQL 与 SQL-92 向后兼容。但是，QLDB 仅支持这些保留字子集。有关 QLDB 当前支持的 SQL 关键字列表，请参见以下主题：
>
> - PartiQL 函数
> - PartiQL 运算符
> - PartiQL 命令

```
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
```

Amazon QLDB 中的 Amazon Ion 数据格式参考

Amazon QLDB 使用一种数据表示法模型，该模型将 Amazon Ion 与一部分 PartiQL类型的子集统一起来。本节提供了 Ion 文档数据格式的参考概述，其与 PartiQL 的集成是分开的。

在 Amazon QLDB 中使用 PartiQL 查询 Ion

有关在 QLDB 中使用 PartiQL 查询 Ion 数据的语法和语义，请参阅 Amazon QLDB PartiQL 参考中的使用 PartiQL 查询 Ion。

有关在 QLDB 分类账中查询和处理 Ion 数据的代码示例，请参阅和Amazon Ion 代码示例和 使用 Amazon Ion。

主题

- 什么是 Amazon Ion？
- Ion 规格
- 兼容 JSON
- JSON 扩展
- Ion 文本示例
- API 参考
- QLDB 中的 Amazon Ion 代码示例

什么是 Amazon Ion？

Ion 是一种开源、类型丰富、自我描述的分层数据序列化格式，最初由 Amazon 内部开发。它基于抽象数据模型，允许您存储结构化和非结构化数据。它是 JSON 的超集，这意味着任何有效的 JSON 文档也是有效的 Ion 文档。本指南假设有 JSON 的基准理论。如果您还不熟悉 JSON，请参阅 JSON 简介了解更多信息。

你可以用人类可读的文本形式或二进制编码的形式交替标注 Ion 文档。与 JSON 一样，文本表单易于读写，支持快速原型设计。二进制编码在保存、传输和解析方面更紧凑和高效。离子处理器可在两种格式之间进行转码，以表示完全相同的数据结构集，而不会丢失任何数据。此功能允许应用程序针对不同的用例优化其数据处理方式。

> **Note**
>
> Ion 数据模型严格基于值，不支持引用。因此，数据模型可以表示可以嵌套至任意深度的数据层次结构，但不能表示有向图。

Ion 规格

请参阅 Amazon GitHub 网站上的 Ion 规范文档，了解 Ion core 数据类型的完整列表，以及完整的描述和值格式化细节。

为了简化应用程序开发，Amazon Ion 提供了可处理 Ion 数据的客户端库。有关显示处理 Ion 数据的常见用例的通用代码示例，请参阅 GitHub 上的 Amazon Ion Cookbook。

兼容 JSON

与 JSON 类似的是，您可以使用一组原始数据类型和一组递归定义的容器类型编写 Amazon Ion 文档。Ion 包含以下传统 JSON 数据类型：

- `null`: 一个通用的非类型空值（空）。此外，如下一节所述，Ion 支持每种基元类型使用不同的空类型。
- `bool`: 布尔值。
- `string`：Unicode 文本文字。
- `list`：有序的异构值集合。
- `struct`：名称/值对的无序集合。与 JSON 一样，`struct`允许每个名称有多个值，但通常不鼓励这样操作。

JSON 扩展

数字类型

Amazon Ion 将数字严格定义为以下number类型之一，而非模棱两可的 JSON 类型：

- `int`：任意大小的有符号整数。
- `decimal`：任意精度的十进制编码实数。
- `float`：二进制编码的浮点数 (64 位 IEEE)。

解析文档时，Ion 处理器会按以下方式分配数字类型：

- `int`：带有小数点但没有指数的数字(例如，`100200`)。
- `decimal`: 没有指数或小数点的数字(例如，`0.00001`、`200.0`)。
- `float`: 带指数的数字，例如科学记数法或 E 记数法 (例如`2e0`、`3.1e-4`)。

新数据类型

Amazon Ion 添加了以下数据类型：

- `timestamp`：任意精度的日期/时间/时区时刻。
- `symbol`：Unicode 符号原子（例如标识符）。
- `blob`：用户定义编码的二进制数据。
- `clob`：用户定义编码的文本数据。
- `sexp`：具有应用程序定义语义的有序值集合。

Null 类型

除了 JSON 定义的通用空类型外，Amazon Ion 还支持每种原始类型使用不同的空类型。这表明在保持严格的数据类型的同时缺乏价值。

```
null
null.null       // Identical to untyped null
null.bool
```

```
null.int
null.float
null.decimal
null.timestamp
null.string
null.symbol
null.blob
null.clob
null.struct
null.list
null.sexp
```

Ion 文本示例

```
// Here is a struct, which is similar to a JSON object.
{
    // Field names don't always have to be quoted.
    name: "fido",

    // This is an integer.
    age: 7,

    // This is a timestamp with day precision.
    birthday: 2012-03-01T,

    // Here is a list, which is like a JSON array.
    toys: [
        // These are symbol values, which are like strings,
        // but get encoded as integers in binary.
        ball,
        rope
    ],
}
```

API 参考

- ion-go
- ion-java
- ion-js
- ion-python

QLDB 中的 Amazon Ion 代码示例

本节提供了通过在 Amazon QLDB 分类账中读取和写入文档值处理 Amazon Ion 数据的代码示例。代码示例使用 QLDB 驱动程序在分类账运行 PartiQL 语句。这些示例是使用示例应用程序教程，开始使用 Amazon QLDB中示例应用程序的一部分，在AWS示例 GitHub 网站上是开源的。

有关显示处理 Ion 数据的常见用例的通用代码示例，请参阅 GitHub 上的 Amazon Ion Cookbook。

运行代码

每种编程语言的教程代码都可执行以下步骤：

1. 连接至 `vehicle-registration` 示例分类账。
2. 创建名为 `IonTypes` 的文件。
3. 在表格中插入包含单个 `Name` 字段的文档。
4. 对于每种支持的Ion 数据类型：
   a. 使用该数据类型的文字值更新文档的`Name`字段。
   b. 查询表格以获取文档最新版本。
   c. 通过检查的值是否与预期的类型相匹配，验证其值是否保留`Name`了其原始数据类型属性。
5. 删除 `IonTypes`表。

> **Note**
>
> 运行本教程代码之前，您必须创建一个名为的分类账`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()
```

Amazon QLDB API 参考

本章介绍可通过 HTTP、AWS Command Line Interface (AWS CLI) 或 AWS SDK 访问的 Amazon QLDB 的低级 API 操作：

- Amazon QLDB — QLDB 资源管理 API（也称为控制面板）。此 API 仅用于管理分类账资源和非事务数据操作。您可以使用这些操作创建、删除、描述、列出和更新分类账。您还可以通过加密方式验证日记账数据，并导出或流式传输日记账块。
- Amazon QLDB 会话 — QLDB 事务数据 API。您可以使用此 API 在带有 PartiQL 语句的分类账上运行数据事务。

> **Important**
>
> 我们建议使用 QLDB 驱动程序或 QLDB Shell 在分类账上运行数据事务，而不是直接与 QLDB 会话 API 交互。
>
> - 如果您在使用 AWS SDK，请使用 QLDB 驱动程序。该驱动程序在 QLDB 会话数据 API 之上提供了一个高级抽象层，并为您管理`SendCommand`操作。有关信息和支持的编程语言列表，请参阅 驱动程序入门。
> - 如果您正在使用 AWS CLI，请使用 QLDB Shell。Shell 是一个命令行接口，它使用 QLDB 驱动程序与分类账进行交互。有关信息，请参阅 使用 Amazon QLDB Shell (仅限数据 API)。

主题

操作

Amazon QLDB 支持以下操作：

- CancelJournalKinesisStream
- CreateLedger

- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

Amazon QLDB 会话支持以下操作：

- SendCommand

Amazon QLDB

Amazon QLDB 支持以下操作：

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export

- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

CancelJournalKinesisStream

服务：Amazon QLDB

结束指定 Amazon QLDB 日记账流。在取消流式传输前，当前状态必须为ACTIVE。

取消流后，您无法重新启动流。已取消的 QLDB 数据流资源有 7 天的保留期，因此在此限制到期后它们会被自动删除。

请求语法

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

streamId

要取消的 QLDB 日记账流的 UUID（以 Base62 编码的文本表示）。

长度限制：固定长度为 22。

模式：^[A-Za-z-0-9]+$

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "StreamId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

StreamId

取消的 QLDB 日记账流的 UUID（以 Base62 编码的文本表示）。

类型：字符串

长度限制：固定长度为 22。

模式：^[A-Za-z-0-9]+$

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateLedger

服务：Amazon QLDB

在您的 AWS 账户 当前区域中创建新的账本。

请求语法

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

DeletionProtection

指定是否保护分类账不被任何用户删除。如果创建分类账期间未定义，则默认情况下启用该功能（`true`）。

如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 `UpdateLedger` 操作将此参数设置为 `false` 来禁用该功能。

类型：布尔值

必需：否

KmsKey

AWS Key Management Service (AWS KMS) 中的密钥，用于加密账本中的静态数据。有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的静态加密。

使用以下选项之一指定此参数：

- AWS_OWNED_KMS_KEY：使用由您代表自己拥有和管理 AWS 的 AWS KMS 密钥。
- 未定义：默认情况下，使用 AWS 拥有的 KMS 密钥。
- A valid symmetric customer managed KMS key（有效的对称客户托管 KMS 密钥）：在您创建、拥有和管理的账户中使用指定的对称加密 KMS 密钥。

  Amazon QLDB 不支持非对称密钥。有关更多信息，请参阅《 AWS Key Management Service 开发人员指南》中的使用对称和非对称密钥。

要指定客户托管的 KMS 密钥，您可以使用其密钥 ID、Amazon 资源名称（ARN）、别名或别名 ARN。使用别名时，应加上 "alias/" 前缀。要在不同的密钥中指定密钥 AWS 账户，必须使用密钥 ARN 或别名 ARN。

例如：

- 密钥 ID：1234abcd-12ab-34cd-56ef-1234567890ab
- 密钥 ARN：arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
- 别名：alias/ExampleAlias
- 别名 ARN：arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias

有关更多信息，请参阅《 AWS Key Management Service 开发者指南》中的密钥标识符 (KeyId)。

类型：字符串

长度限制：最大长度为 1600。

必需：否

Name

您要创建的分类帐的名称。该名称在您 AWS 账户 当前区域的所有账本中必须是唯一的。

分类账名称的命名约束在《Amazon QLDB 开发人员指南》的 Amazon QLDB 中的配额中定义。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

PermissionsMode

要分配给您想创建的分类帐的权限模式。此参数可能具有下列值之一：

- ALLOW_ALL：一种旧式权限模式，支持对分类账进行 API 级别粒度的访问控制。

  此模式允许拥有此分类账的 SendCommand API 权限的用户在指定分类账中的任何表上运行所有 PartiQL 命令（因此，ALLOW_ALL）。此模式忽略您为分类账创建的任何表级或命令级 IAM 权限策略。

- STANDARD：（推荐）一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。

  默认情况下，此模式拒绝所有用户请求在此分类账中的任何表上运行任何 PartiQL 命令。要允许 PartiQL 命令运行，除了分类账的 SendCommand API 权限外，您还必须为特定表资源和 PartiQL 操作创建 IAM 权限策略。有关信息，请参阅《Amazon QLDB 开发人员指南》中的标准权限模式入门。

> **Note**
>
> 我们强烈建议使用 STANDARD 权限模式来最大限度地提高分类账数据的安全性。

类型：字符串

有效值：ALLOW_ALL | STANDARD

必需：是

Tags

要作为标签添加到待创建的分类账中的键值对。标签键区分大小写。标签值区分大小写，可以为空值。

类型：字符串到字符串映射

映射条目：最低 0 项。最多 200 项。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最小长度为 0。最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Arn

分类账的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

CreationDateTime

创建分类账时，日期和时间采用世界标准纪元时间格式）。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

类型：时间戳

DeletionProtection

指定是否保护分类账不被任何用户删除。如果创建分类账期间未定义，则默认情况下启用该功能（`true`）。

如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 `UpdateLedger` 操作将此参数设置为 `false` 来禁用该功能。

类型：布尔值

KmsKeyArn

分类账用于静态加密的、客户托管 KMS 密钥。如果未定义此参数，则账本使用 AWS 拥有的 KMS 密钥进行加密。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

Name

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

PermissionsMode

您想创建的分类帐的权限模式。

类型：字符串

有效值：ALLOW_ALL | STANDARD

State

分类账的当前状态。

类型：字符串

有效值：CREATING | ACTIVE | DELETING | DELETED

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

LimitExceededException

您已达到允许的最大资源数量上限。

HTTP 状态代码：400

ResourceAlreadyExistsException

指定资源已经存在。

HTTP 状态代码：409

ResourceInUseException

此时无法修改指定资源。

HTTP 状态代码：409

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteLedger

服务：Amazon QLDB

删除分类账及其所有内容。此操作不可逆。

如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 UpdateLedger 操作将此参数设置为 false 来禁用该功能。

请求语法

```
DELETE /ledgers/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

您要删除的日记账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
```

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceInUseException

此时无法修改指定资源。

HTTP 状态代码：409

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeJournalKinesisStream

服务：Amazon QLDB

返回有关给定 Amazon QLDB 日记账流的详细信息。输出包括 Amazon 资源名称（ARN）、流名称、当前状态、创建时间和原始流创建请求参数。

此操作不会返回任何过期的日志流。有关更多信息，请参阅 《Amazon QLDB 开发人员指南》 中的 终端流过期。

请求语法

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

streamId

QLDB 日记账流描述的 UUID（以 Base62 编码的文本表示）。

长度限制：固定长度为 22。

模式：^[A-Za-z-0-9]+$

必需：是

请求体

该请求没有请求正文。

响应语法

```
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"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Stream

DescribeJournalS3Export 请求返回的 QLDB 日记账流信息。

类型：JournalKinesisStreamDescription 对象

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeJournalS3Export

服务：Amazon QLDB

有关日记账导出任务的返回信息，包括分类账名称、导出 ID、创建时间、当前状态和原始导出创建请求的参数。

此操作不会返回任何过期导出作业。有关更多信息，请参阅《Amazon QLDB 开发者指南》中的过期导出作业。

如果给定的导出任务`ExportId`不存在，则抛出`ResourceNotFoundException`。

如果给定`Name`分类账不存在，则抛出`ResourceNotFoundException`。

请求语法

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

exportId

描述的 QLDB 日记账导出作业的 UUID（以 Base62 编码的文本表示）。

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：是

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：是

请求体

该请求没有请求正文。

响应语法

```
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"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

ExportDescription

`DescribeJournalS3Export` 请求返回的日记账导出作业信息。

类型：JournalS3ExportDescription 对象

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版 SDK
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeLedger

服务：Amazon QLDB

返回有关分类账的信息，包括分类账状态、权限模式、静态加密设置以及创建时间。

请求语法

```
GET /ledgers/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

您要描述的分类帐的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

该请求没有请求正文。

响应语法

```
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"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Arn

分类账的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

CreationDateTime

创建分类账时，日期和时间采用世界标准纪元时间格式）。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

类型：时间戳

DeletionProtection

指定是否保护分类账不被任何用户删除。如果创建分类账期间未定义，则默认情况下启用该功能（`true`）。

如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 `UpdateLedger` 操作将此参数设置为 `false` 来禁用该功能。

类型：布尔值

EncryptionDescription

有关分类账中静态数据加密的信息。这包括当前状态、 AWS KMS 密钥以及何时无法访问密钥（如果出现错误）。如果未定义此参数，则账本使用 AWS 拥有的 KMS 密钥进行加密。

类型：LedgerEncryptionDescription 对象

Name

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

PermissionsMode

分类账的权限模式。

类型：字符串

有效值：ALLOW_ALL | STANDARD

State

分类账的当前状态。

类型：字符串

有效值：CREATING | ACTIVE | DELETING | DELETED

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ExportJournalToS3

服务：Amazon QLDB

将日期和时间范围内的日记账内容从分类账导出至 Amazon Simple Storage Service（Amazon S3）存储桶中。日记账导出任务可以以 Amazon Ion 格式的文本或二进制形式或 JSON 行 文本格式写入数据对象。

如果给定Name分类账不存在，则抛出ResourceNotFoundException。

如果给定Name分类账处于CREATING状态，则抛出ResourcePreconditionNotMetException。

您最多可为每个分类帐启动两个并发日记账账导出请求。超过此限制，则日记账账导出请求会抛出LimitExceededException。

请求语法

```
POST /ledgers/name/journal-s3-exports HTTP/1.1
Content-type: application/json

{
   "ExclusiveEndTime": number,
   "InclusiveStartTime": number,
   "OutputFormat": "string",
   "RoleArn": "string",
   "S3ExportConfiguration": {
      "Bucket": "string",
      "EncryptionConfiguration": {
         "KmsKeyArn": "string",
         "ObjectEncryptionType": "string"
      },
      "Prefix": "string"
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

ExclusiveEndTime

要导出的日记账内容范围的唯一结束日期和时间。

ExclusiveEndTime 必须采用 ISO 8601 日期和时间格式以及通用协调时间（UTC）。例如：2019-06-13T21:36:34Z。

ExclusiveEndTime 必须小于或等于当前的 UTC 日期和时间。

类型：时间戳

必需：是

InclusiveStartTime

要导出的日记账内容范围的全部起始日期和时间。

InclusiveStartTime 必须采用 ISO 8601 日期和时间格式以及通用协调时间（UTC）。例如：2019-06-13T21:36:34Z。

InclusiveStartTime必须在ExclusiveEndTime之前。

如果您提供的 InclusiveStartTime 是在分类账的 CreationDateTime 之前，则 Amazon QLDB 有效地将其默认视为分类账的 CreationDateTime。

类型：时间戳

必需：是

OutputFormat

导出的日记账数据的输出格式。日记账导出作业可以以 Amazon Ion 格式的文本或二进制形式或 JSON 行文本格式写入数据对象。

默认：ION_TEXT

在 JSON 行格式中，导出的数据对象中的每个日记账数据块都是由换行符分隔的有效 JSON 对象。您可以使用这种格式将 JSON 导出与 Amazon Athena 和 AWS Glue 等分析工具直接集成，因为这些服务可以自动解析以换行符分隔的 JSON。

类型：字符串

有效值：`ION_BINARY | ION_TEXT | JSON`

必需：否

RoleArn

IAM 角色的 Amazon 资源名称（ARN），该名称授予日记账导出任务的 QLDB 权限：

- 将对象写入 Amazon S3 存储桶。
- （可选）使用 AWS Key Management Service (AWS KMS) 中的客户托管密钥对导出的数据进行服务器端加密。

要在请求日记账导出时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 `iam:PassRole` 操作的权限。这是所有日记账流请求所必需的。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：是

S3ExportConfiguration

导出请求 Amazon S3 存储桶目标的配置设置。

类型：S3ExportConfiguration 对象

必需：是

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

ExportId

QLDB 分配到每个日记账流导出作业的 UUID（以 Base62 编码的文本表示）。

要描述您的导出请求并检查任务状态，您可以使用`ExportId`致电`DescribeJournalS3Export`。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK

- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

GetBlock

服务：Amazon QLDB

返回日志中指定地址的数据块对象。如果 DigestTipAddress 已提供，还会返回指定数据块的证明以供验证。

有关区块中数据内容的信息，请参阅 Amazon QLDB 开发人员指南中的日志内容。

如果指定的分类账不存在或处于DELETING状态，则抛出ResourceNotFoundException。

如果指定的分类账处于CREATING状态，则抛出ResourcePreconditionNotMetException。

如果不存在具有指定地址的数据块，则抛出InvalidParameterException。

请求语法

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

BlockAddress

您要请求的数据块的位置。地址是一种包含两个字段的 Amazon Ion 结构：即 `strandId` 和 `sequenceNo`。

例如：`{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`。

类型：ValueHolder 对象

必需：是

DigestTipAddress

摘要中要求提供证据的最新数据块位置。地址是一种包含两个字段的 `strandId`Amazon Ion 结构：即 和 `sequenceNo`。

例如：`{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`。

类型：ValueHolder 对象

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Block

Amazon Ion 格式的区块数据对象。

类型：ValueHolder 对象

Proof

GetBlock 请求返回的 Amazon Ion 格式的证明对象。证明包含使用默克尔树重新计算指定摘要所需哈希值列表，从指定的区块开始。

类型：ValueHolder 对象

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

GetDigest

服务：Amazon QLDB

返回日记账中最新提交块的分类账摘要。响应包括一个 256 位的哈希值和一个块地址。

请求语法

```
POST /ledgers/name/digest HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Digest

由 `GetDigest` 代表请求返回的摘要的 256 位哈希值。

类型：Base64 编码的二进制数据对象

长度限制：固定长度为 32。

DigestTipAddress

摘要中要求提供证据的最新数据块位置。地址是一种包含两个字段的 `strandId`Amazon Ion 结构：即 和 `sequenceNo`。

类型：ValueHolder 对象

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面

- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

GetRevision

服务：Amazon QLDB

返回指定文档 ID 和块地址的修订数据对象。如果 `DigestTipAddress` 已提供，还会返回指定修订版的证明以供验证。

请求语法

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

BlockAddress

要验证的文档修订版的数据块位置。地址是一种包含两个字段的 `strandId`Amazon Ion 结构：即和 `sequenceNo`。

例如：{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}。

类型：ValueHolder 对象

必需：是

DigestTipAddress

摘要中要求提供证据的最新数据块位置。地址是一种包含两个字段的 strandIdAmazon Ion 结构：即 和 sequenceNo。

例如：{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}。

类型：ValueHolder 对象

必需：否

DocumentId

要验证的文档的 UUID（以 Base62 编码的文本表示）。

类型：字符串

长度限制：固定长度为 22。

模式：^[A-Za-z-0-9]+$

必需：是

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
   },
   "Revision": {
      "IonText": "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Proof

GetRevision 请求返回的 Amazon Ion 格式的证明对象。证明包含使用默克尔树重新计算指定摘要所需哈希值列表，从指定的文档修订版开始。

类型：ValueHolder 对象

Revision

Amazon Ion 格式的文档修订数据对象。

类型：ValueHolder 对象

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListJournalKinesisStreamsForLedger

服务：Amazon QLDB

返回给定分类账的所有 Amazon QLDB 日记账流。

此操作不会返回任何过期的日记账流。有关更多信息，请参阅 《Amazon QLDB 开发人员指南》 中的终端流过期。

此操作返回最多 `MaxResults` 个项目。它是分页的，因此您可以通过 `ListJournalKinesisStreamsForLedger` 多次调用来检索所有项目。

请求语法

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：是

MaxResults

在单个 `ListJournalKinesisStreamsForLedger` 请求中要返回的最大结果数。（返回的实际结果数可能会更少。）

有效范围：最小值为 1。最大值为 100。

NextToken

分页令牌表示您要检索下一页结果。如果您在上次`ListJournalKinesisStreamsForLedger`调用的响应中收到了 `NextToken` 的值，则应使用 值作为此处的输入。

长度限制：最小长度为 4。长度上限为 1024。

模式：`^[A-Za-z-0-9+/=]+$`

请求正文

该请求没有请求正文。

响应语法

```
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"
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

NextToken

- 如果 NextToken 为空，则表示最后一页结果已处理完毕，没有检索到其他结果。
- 如果NextToken 不 为空，则表示有更多结果可用。若要检索下一页结果，请在后续 ListJournalKinesisStreamsForLedger 调用中使用NextToken值。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

Streams

当前与给定分类账关联的 QLDB 日记账流。

类型：JournalKinesisStreamDescription 对象数组

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包

- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListJournalS3Exports

服务：Amazon QLDB

返回与当前 AWS 账户 和区域关联的所有分类账的所有日记账导出任务。

此操作最多返回 `MaxResults` 个项目，并且会进行分页，以便您可以通过 `ListJournalS3Exports` 个多次调用来检索所有项目。

此操作不会返回任何过期导出作业。有关更多信息，请参阅 《Amazon QLDB 开发者指南》 中的 导出作业到期。

请求语法

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

在单个 `ListJournalS3Exports` 请求中要返回的最大结果数。（返回的实际结果数可能会更少。）

有效范围：最小值为 1。最大值为 100。

NextToken

分页令牌表示您要检索下一页结果。如果您在上次 `ListJournalS3Exports` 调用的响应中收到了 `NextToken` 的值，则应使用该值作为此处的输入。

长度限制：最小长度为 4。长度上限为 1024。

模式：`^[A-Za-z-0-9+/=]+$`

请求正文

该请求没有请求正文。

响应语法

```
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"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

JournalS3Exports

返回与当前 AWS 账户 和区域关联的所有分类账的日记账导出作业。

类型：JournalS3ExportDescription 对象数组

NextToken

- 如果 NextToken 为空，则表示最后一页结果已处理完毕，没有其他结果。
- 如果NextToken 不 为空，则表示有更多结果可用。若要检索下一页结果，请在后续 ListJournalS3Exports 调用中使用NextToken值。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

错误

有关所有操作返回的常见错误的信息，请参阅 常见错误。

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListJournalS3ExportsForLedger

服务：Amazon QLDB

返回指定分类账的所有日记账导出作业。

此操作最多返回 MaxResults 个项目，并且会进行分页，以便您可以通过 ListJournalS3ExportsForLedger 个多次调用来检索所有项目。

此操作不会返回任何过期导出作业。有关更多信息，请参阅 《Amazon QLDB 开发者指南》 中的 导出作业到期。

请求语法

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

在单个 ListJournalS3ExportsForLedger 请求中要返回的最大结果数。（返回的实际结果数可能会更少。）

有效范围：最小值为 1。最大值为 100。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

NextToken

分页令牌表示您要检索下一页结果。如果您在上次 ListJournalS3ExportsForLedger 调用的响应中收到了 NextToken 的值，则应使用该值作为此处的输入。

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

请求正文

该请求没有请求正文。

响应语法

```
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"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

JournalS3Exports

当前与指定分类账关联的日记账导出任务。

类型：JournalS3ExportDescription 对象数组

NextToken

- 如果 NextToken 为空，则表示最后一页结果已处理完毕，没有其他结果。
- 如果NextToken 不 为空，则表示有更多结果可用。若要检索下一页结果，请在后续 ListJournalS3ExportsForLedger 调用中使用NextToken值。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

错误

有关所有操作返回的常见错误的信息，请参阅 常见错误。

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版 SDK
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListLedgers

服务：Amazon QLDB

返回与当前 AWS 账户 和地区关联的所有分类账。

此操作最多返回 MaxResults 个项目，并且会进行分页，以便您可以通过 ListLedgers 个多次调用来检索所有项目。

请求语法

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

在单个 ListLedgers 请求中要返回的最大结果数。（返回的实际结果数可能会更少。）

有效范围：最小值为 1。最大值为 100。

NextToken

分页令牌表示您要检索下一页结果。如果您在上次 ListLedgers 调用的响应中收到了 NextToken 的值，则应使用该值作为此处的输入。

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Ledgers": [
```

```
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Ledgers

与当前 AWS 账户 和地区关联的分类账。

类型：LedgerSummary 对象数组

NextToken

分页标记，表示是否还有更多结果可用：

- 如果 NextToken 为空，则表示最后一页结果已处理完毕，不需要检索其他结果。
- 如果NextToken 不 为空，则表示有更多结果可用。若要检索下一页结果，请在后续 ListLedgers 调用中使用NextToken值。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

错误

有关所有操作返回的常见错误的信息，请参阅 常见错误。

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面

- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListTagsForResource

服务：Amazon QLDB

为指定的 Amazon QLDB 资源返回所有标签。

请求语法

```
GET /tags/resourceArn HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

resourceArn

要列出其标签的Amazon 资源名称（ARN）。例如：

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

长度约束：最小长度为 20。长度上限为 1600。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Tags

当前与指定的 Amazon QLDB 资源关联的标签。

类型：字符串到字符串映射

映射条目：最低 0 项。最多 200 项。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最小长度为 0。最大长度为 256。

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3

- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

StreamJournalToKinesis

服务：Amazon QLDB

为给定的 Amazon QLDB 分类账创建日记账流。流捕获提交到分类账的日志的每个文档修订版本，并将数据传送到指定的 Amazon Kinesis Data Streams 资源。

请求语法

```
POST /ledgers/name/journal-kinesis-streams HTTP/1.1
Content-type: application/json

{
   "ExclusiveEndTime": number,
   "InclusiveStartTime": number,
   "KinesisConfiguration": {
      "AggregationEnabled": boolean,
      "StreamArn": "string"
   },
   "RoleArn": "string",
   "StreamName": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

ExclusiveEndTime

指定流结束时间的日期和时间（不内含）。如果不定义此参数，则流将无限期运行，直到您取消它。

ExclusiveEndTime 必须采用 ISO 8601 日期和时间格式以及通用协调时间（UTC）。例如：2019-06-13T21:36:34Z。

类型：时间戳

必需：否

InclusiveStartTime

开始流式传输日记账数据的开始日期和时间（内含）。此参数必须采用 ISO 8601 日期和时间格式以及通用协调时间 (UTC)。例如：2019-06-13T21:36:34Z。

InclusiveStartTime 不能是未来时间，必须在 ExclusiveEndTime 之前。

如果您提供的 InclusiveStartTime 是在分类账的 CreationDateTime 之前，则 QLDB 有效地将其默认视为分类账的 CreationDateTime。

类型：时间戳

必需：是

KinesisConfiguration

流请求的 Kinesis 数据流目标的配置设置。

类型：KinesisConfiguration 对象

必需：是

RoleArn

IAM 角色的 Amazon 资源名称（ARN），该角色授予日记账流将数据记录写入 Kinesis 数据流资源的 QLDB 权限。

要在请求日志流时将角色传递给 QLDB，您必须具有对 IAM 角色资源执行 iam:PassRole 操作的权限。这是所有日志流请求所必需的。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：是

StreamName

要分配给 QLDB 日记账流的名称。用户定义的名称有助于识别和指示流的用途。

您的流名称在给定分类账的其他活动的 流中必须是唯一的。流名称与分类账名称具有相同的命名约束，如在《Amazon QLDB 开发人员指南》的 Amazon QLDB 中的配额中所定义。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：是

Tags

要作为标签添加到待创建的流中的键值对。标签键区分大小写。标签值区分大小写，可以为空值。

类型：字符串到字符串映射

映射条目：最低 0 项。最多 200 项。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最小长度为 0。最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

StreamId

QLDB 分配到每个 QLDB 日记账流的 UUID（以 Base62 编码的文本表示）。

类型：字符串

长度限制：固定长度为 22。

模式：^[A-Za-z-0-9]+$

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

ResourcePreconditionNotMetException

由于未提前满足条件，操作失败。

HTTP 状态代码：412

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK

- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

TagResource

服务：Amazon QLDB

将一个或多个标签添加到指定的 Amazon QLDB 资源。

资源最多可以包含 50 个标签。如果尝试为资源创建超过 50 个标签，则请求将失败并返回错误。

请求语法

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

resourceArn

您希望为其添加标签的 Amazon 资源名称（ARN）。例如：

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

长度约束：最小长度为 20。长度上限为 1600。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

Tags

要添加为标签的键值对将以标签形式添加到指定 QLDB 资源。标签键区分大小写。如果为资源指定的密钥已经存在，则请求将失败并返回错误。标签值区分大小写，可以为空值。

类型：字符串到字符串映射

映射条目：最低 0 项。最多 200 项。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最小长度为 0。最大长度为 256。

必需：是

响应语法

```
HTTP/1.1 200
```

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UntagResource

服务：Amazon QLDB

删除指定 Amazon QLDB 资源的一个或多个标签。您最多可指定 50 个要删除的标签键。

请求语法

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

resourceArn

要从中删除标签的 Amazon 资源名称（ARN）。例如：

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

长度约束：最小长度为 20。长度上限为 1600。

必需：是

TagKeys

要删除的标签键列表。

数组成员：最少 0 个物品。最多 200 项。

长度限制：长度下限为 1。长度上限为 128。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
```

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateLedger

服务：Amazon QLDB

更新分类账上的属性。

请求语法

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

DeletionProtection

指定是否保护分类账不被任何用户删除。如果创建分类账期间未定义，则默认情况下启用该功能（`true`）。

如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 `UpdateLedger` 操作将此参数设置为 `false` 来禁用该功能。

类型：布尔值

必需：否

KmsKey

AWS Key Management Service (AWS KMS) 中的密钥，用于加密账本中的静态数据。有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的静态加密。

使用以下选项之一指定此参数：

- AWS_OWNED_KMS_KEY：使用由您代表自己拥有和管理 AWS 的 AWS KMS 密钥。
- 未定义：不对分类账的 KMS 密钥进行任何更改。
- A valid symmetric customer managed KMS key（有效的对称客户托管 KMS 密钥）：在您创建、拥有和管理的账户中使用指定的对称加密 KMS 密钥。

  Amazon QLDB 不支持非对称密钥。有关更多信息，请参阅《 AWS Key Management Service 开发人员指南》中的使用对称和非对称密钥。

要指定客户托管的 KMS 密钥，您可以使用其密钥 ID、Amazon 资源名称（ARN）、别名或别名 ARN。使用别名时，应加上 "alias/" 前缀。要在不同的密钥中指定密钥 AWS 账户，必须使用密钥 ARN 或别名 ARN。

例如：

- 密钥 ID：1234abcd-12ab-34cd-56ef-1234567890ab
- 密钥 ARN：arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
- 别名：alias/ExampleAlias
- 别名 ARN：arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias

有关更多信息，请参阅《 AWS Key Management Service 开发者指南》中的密钥标识符 (KeyId)。

类型：字符串

长度限制：最大长度为 1600。

必需：否

响应语法

```
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"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Arn

分类账的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

CreationDateTime

创建分类账时，日期和时间采用世界标准纪元时间格式）。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

类型：时间戳

DeletionProtection

指定是否保护分类账不被任何用户删除。如果创建分类账期间未定义，则默认情况下启用该功能（`true`）。

如果启用了删除保护，您必须先禁用它，然后才能删除分类账。您可以通过调用 `UpdateLedger` 操作将此参数设置为 `false` 来禁用该功能。

类型：布尔值

EncryptionDescription

有关分类账中静态数据加密的信息。这包括当前状态、 AWS KMS 密钥以及何时无法访问密钥（如果出现错误）。

类型：LedgerEncryptionDescription 对象

Name

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

分类账的当前状态。

类型：字符串

有效值：`CREATING | ACTIVE | DELETING | DELETED`

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateLedgerPermissionsMode

服务：Amazon QLDB

更新分类账的权限模式。

> **Important**
>
> 在切换至 STANDARD 权限模式前，必须先创建所有必需的 IAM 策略和表标签，以避免对用户造成干扰。要了解更多信息，请参阅 Amazon QLDB 开发人员指南中的迁移到标准权限模式。

请求语法

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

URI 请求参数

请求使用以下 URI 参数。

name

分类账的名称。

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

PermissionsMode

为您的分类账分配权限的模式。此参数可能具有下列值之一：

- ALLOW_ALL：一种旧式权限模式，支持对分类账进行 API 级别粒度的访问控制。

  此模式允许拥有此分类账的 SendCommand API 权限的用户在指定分类账中的任何表上运行所有 PartiQL 命令（因此，ALLOW_ALL）。此模式忽略您为分类账创建的任何表级或命令级 IAM 权限策略。

- STANDARD：（推荐）一种权限模式，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。

  默认情况下，此模式拒绝所有用户请求在此分类账中的任何表上运行任何 PartiQL 命令。要允许 PartiQL 命令运行，除了分类账的 SendCommand API 权限外，您还必须为特定表资源和 PartiQL 操作创建 IAM 权限策略。有关信息，请参阅《Amazon QLDB 开发人员指南》中的标准权限模式入门。

> **Note**
>
> 我们强烈建议使用 STANDARD 权限模式来最大限度地提高分类账数据的安全性。

类型：字符串

有效值：ALLOW_ALL | STANDARD

必需：是

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "Name": "string",
   "PermissionsMode": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Arn

分类账的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

Name

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

PermissionsMode

分类账的当前权限模式。

类型：字符串

有效值：ALLOW_ALL | STANDARD

错误

有关所有操作返回的常见错误的信息，请参阅常见错误。

InvalidParameterException

请求中的一项或多项参数无效。

HTTP 状态代码：400

ResourceNotFoundException

指定的资源不存在。

HTTP 状态代码：404

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

Amazon QLDB 会话

Amazon QLDB 会话支持以下操作：

- SendCommand

SendCommand

服务：Amazon QLDB Session

向 Amazon QLDB 分类账发送命令。

> **Note**
>
> 我们建议使用 QLDB 驱动程序或 QLDB Shell 在分类账上运行数据事务，而不是直接与此 API 交互。
>
> - 如果您使用的是 AWS 软件开发工具包，请使用 QLDB 驱动程序。该驱动程序在 QLDB 会话数据 API 之上提供了一个高级抽象层，并为您管理SendCommand操作。有关信息和支持的编程语言列表，请参阅 Amazon QLDB 开发者指南中的驱动程序入门。
> - 如果您正在使用 AWS Command Line Interface (AWS CLI)，请使用 QLDB 外壳。Shell 是一个命令行接口，它使用 QLDB 驱动程序与分类账进行交互。有关信息，请参阅使用 QLDB Shell 访问 Amazon QLDB。

请求语法

```
{
   "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": {
   }
}
```

请求参数

有关所有操作的通用参数的信息，请参阅通用参数。

请求接受采用 JSON 格式的以下数据。

AbortTransaction

命令中止当前事务。

类型：AbortTransactionRequest 对象

必需：否

CommitTransaction

命令提交指定的事务。

类型：CommitTransactionRequest 对象

必需：否

EndSession

命令结束当前会话。

类型：EndSessionRequest 对象

必需：否

ExecuteStatement

命令在指定事务中执行语句。

类型：ExecuteStatementRequest 对象

必需：否

FetchPage

命令获取页面。

类型：FetchPageRequest 对象

必需：否

SessionToken

指定当前命令的会话令牌。会话令牌在会话的整个生命周期中都是恒定的。

要获取会话令牌，请运行 StartSession 命令。这个 SessionToken 对于当前会话期间发出的每个后续命令都是必需的。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

必需：否

StartSession

命令启动新会话。作为响应的一部分获得会话令牌。

类型：StartSessionRequest 对象

必需：否

StartTransaction

命令启动新事务。

类型：StartTransactionRequest 对象

必需：否

响应语法

```
{
   "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"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

AbortTransaction

包含已中止事务的详细信息。

类型：AbortTransactionResult 对象

CommitTransaction

包含已提交事务的详细信息。

类型：CommitTransactionResult 对象

EndSession

包含已结束会话的详细信息。

类型：EndSessionResult 对象

ExecuteStatement

包含已执行语句的详细信息。

类型：ExecuteStatementResult 对象

FetchPage

包含获取的页面的详细信息。

类型：FetchPageResult 对象

StartSession

包含已启动会话的详细信息，其中包括会话令牌。这个 SessionToken 对于当前会话期间发出的每个后续命令都是必需的。

类型：StartSessionResult 对象

StartTransaction

包含已启动事务的详细信息。

类型：StartTransactionResult 对象

错误

有关所有操作的常见错误信息，请参阅常见错误。

BadRequestException

如果请求格式错误或包含错误（例如参数值无效或缺少必填参数），则返回。

HTTP 状态代码：400

CapacityExceededException

当请求超出分类账的处理能力时返回。

HTTP 状态代码：400

InvalidSessionException

如果会话因超时或过期而不再存在，则返回。

HTTP 状态代码：400

LimitExceededException

如果超过资源限制（例如活动会话数），则返回。

HTTP 状态代码：400

OccConflictException

OccConflictException – 当由于乐观并发控制（OCC）的验证阶段失败而导致事务无法写入日记账时返回。

HTTP 状态代码：400

RateExceededException

请求速率超出允许吞吐量时返回。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

数据类型

Amazon QLDB 支持以下数据类型：

- JournalKinesisStreamDescription
- JournalS3ExportDescription

- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

Amazon QLDB 会话支持以下数据类型：

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

Amazon QLDB

Amazon QLDB 支持以下数据类型：

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

JournalKinesisStreamDescription

服务：Amazon QLDB

有关 Amazon QLDB 日记账流的信息，包括 Amazon 资源名称（ARN）、流名称、创建时间、当前状态和原始流创建请求参数。

内容

KinesisConfiguration

Amazon QLDB 日记账流的 Amazon Kinesis Data Streams 目标的配置设置。

类型：KinesisConfiguration 对象

必需：是

LedgerName

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：是

RoleArn

IAM 角色的 Amazon 资源名称（ARN），该角色授予日记账流将数据记录写入 Kinesis 数据流资源的 QLDB 权限。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：是

Status

QLDB 日记账流当前状态。

类型：字符串

有效值：ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED

必需：是

StreamId

QLDB 日记账流的 UUID（以 Base62 编码的文本表示）。

类型：字符串

长度限制：固定长度为 22。

模式：^[A-Za-z-0-9]+$

必需：是

StreamName

QLDB 日记账流的用户定义名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

Arn

QLDB 日记账流的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：否

CreationTime

QLDB 日记账流创建时的日期和时间，采用纪元时间格式。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

类型：时间戳

必需：否

ErrorCause

关于说明数据流状态IMPAIRED 或 FAILED原因的错误消息。这不适用于包含其他状态值的数据流。

类型：字符串

有效值：KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED

必需：否

ExclusiveEndTime

指定流结束时间的日期和时间（不内含）。如果不定义此参数，则流将无限期运行，直到您取消它。

类型：时间戳

必需：否

InclusiveStartTime

开始流式传输日记账数据的开始日期和时间（内含）。

类型：时间戳

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

JournalS3ExportDescription

服务：Amazon QLDB

有关日记账导出任务的信息，包括分类账名称、导出 ID、创建时间、当前状态和原始导出创建请求的参数。

内容

ExclusiveEndTime

原始导出请求中指定的日记账内容范围的唯一结束日期和时间。

类型：时间戳

必需：是

ExportCreationTime

创建导出作业时，日期和时间采用世界标准纪元时间格式。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

类型：时间戳

必需：是

ExportId

日记账流作业的 UUID（以 Base62 编码的文本表示）。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：是

InclusiveStartTime

包含在原始导出请求中指定的日记账内容范围的开始日期和时间。

类型：时间戳

必需：是

LedgerName

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$

必需：是

RoleArn

IAM 角色的 Amazon 资源名称（ARN），该角色为日记账导出任务授予 QLDB 权限，以执行如下操作：

- 将对象写入 Amazon Simple Storage Service（Amazon S3）存储桶。
- （可选）使用 AWS Key Management Service (AWS KMS) 中的客户托管密钥对导出的数据进行服务器端加密。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：是

S3ExportConfiguration

Amazon Simple Storage Service（Amazon S3）存储桶的位置，日记账导出作业会在其中写入日记账内容。

类型：S3ExportConfiguration 对象

必需：是

Status

日记账导出作业的当前状态。

类型：字符串

有效值：IN_PROGRESS | COMPLETED | CANCELLED

必需：是

OutputFormat

导出的日记账数据输出格式。

类型：字符串

有效值：`ION_BINARY | ION_TEXT | JSON`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

KinesisConfiguration

服务：Amazon QLDB

Amazon QLDB 日志流的 Amazon Kinesis Data Streams 目标的配置设置。

内容

StreamArn

请记下 Kinesis 数据流资源的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：是

AggregationEnabled

使 QLDB 能够在单个 Kinesis 数据流记录中发布多个数据记录，从而增加每个 API 调用发送的记录数量。

默认：`True`

> **Important**
>
> 记录聚合对记录处理具有重要影响，并且需要在流使用者中取消聚合。要了解更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的 KPL 主要概念以及使用者取消聚合。

类型：布尔值

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS 适用于 Ruby V3 的 SDK

LedgerEncryptionDescription

服务：Amazon QLDB

有关 Amazon QLDB 分类账中静态数据加密的信息。这包括当前状态、 AWS Key Management Service (AWS KMS) 中的密钥以及密钥何时变得无法访问（如果出现错误）。

有关更多信息，请参阅《Amazon QLDB 开发人员指南》中的静态加密。

内容

EncryptionStatus

分类账静态加密的当前状态。它可以是以下值之一：

- ENABLED：使用指定的密钥完全启用加密。
- UPDATING：分类账正在积极处理指定的密钥更改。

  QLDB 中的关键更改为异步类型。在处理关键变更期间，分类账完全可以访问，不会对性能产生任何影响。秘钥更新所用时间因分类账大小而异。
- KMS_KEY_INACCESSIBLE：无法访问指定的客户托管的 KMS 密钥，并且分类账受损。密钥被禁用或删除，或对密钥的授权已被撤销。当分类账受损时，它将无法访问，并且不接受任何读取或写入请求。

  在您恢复对密钥的授权或重新启用已禁用的密钥之后，受损分类账会自动返回至活动状态。但是，删除客户托管的 KMS 密钥操作不可逆。删除密钥后，您无法再访问使用该密钥保护的分类帐，并且该数据将无法永久恢复。

类型：字符串

有效值：ENABLED | UPDATING | KMS_KEY_INACCESSIBLE

必需：是

KmsKeyArn

ARN — 分类账用于静态加密的、客户托管 KMS 密钥的 Amazon 资源名称（ARN）。如果未定义此参数，则账本使用 AWS 拥有的 KMS 密钥进行加密。AWS_OWNED_KMS_KEY当将账本的加密配置更新为所 AWS 拥有的 KMS 密钥时，它将显示。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：是

InaccessibleKmsKeyDateTime

出现错误时 AWS KMS 密钥首次变得无法访问的日期和时间，采用纪元时间格式。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

如果 AWS KMS 密钥可访问，则此参数未定义。

类型：时间戳

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

LedgerSummary

服务：Amazon QLDB

有关分类账的信息，包括分类账的名称、状态和创建时间。

内容

CreationDateTime

创建分类账时，日期和时间采用世界标准纪元时间格式）。（纪元时间格式为自 1970 年 1 月 1 日午夜 12:00:00 UTC 以来的秒数。）

类型：时间戳

必需：否

Name

分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：否

State

分类账的当前状态。

类型：字符串

有效值：`CREATING | ACTIVE | DELETING | DELETED`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS 适用于 Ruby V3 的 SDK

S3EncryptionConfiguration

服务：Amazon QLDB

日记账导出作业在 Amazon Simple Storage Service（Amazon S3）存储桶中写入数据时使用的加密设置。

内容

ObjectEncryptionType

Amazon S3 对象加密类型。

有关在 Amazon S3 中使用服务器端加密选项的信息，请参阅 Amazon S3 开发人员指南 中的使用服务器端加密保护数据。

类型：字符串

有效值：`SSE_KMS | SSE_S3 | NO_ENCRYPTION`

必需：是

KmsKeyArn

() 中对称加密密钥的亚马逊资源名称 (ARN)。 AWS Key Management Service AWS KMS Amazon S3 不支持非对称 KMS 密钥。

如果您指定 `SSE_KMS` 为 `ObjectEncryptionType`，则必须提供 `KmsKeyArn`。

如果指定 `SSE_S3` 作为 `ObjectEncryptionType`，则 `KmsKeyArn` 为必需。

类型：字符串

长度约束：最小长度为 20。长度上限为 1600。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS 适用于 Ruby V3 的 SDK

S3ExportConfiguration

服务：Amazon QLDB

Amazon Simple Storage Service（Amazon S3）存储桶的位置，日记账导出作业会在其中写入日记账内容。

内容

Bucket

Amazon S3 存储桶的名称，日记账导出作业会在其中写入日记账内容。

存储桶名称必须符合 Amazon S3 存储桶命名约定。有关更多信息，请参阅《Amazon S3 用户指南》中的存储桶约束和限制。

类型：字符串

长度约束：最小长度为 3。最大长度为 255。

模式：`^[A-Za-z-0-9-_.]+$`

必需：是

EncryptionConfiguration

日记账导出任务用于在 Amazon S3 存储桶中写入数据的加密设置。

类型：S3EncryptionConfiguration 对象

必需：是

Prefix

日记账导出作业写入日记账内容的 Amazon S3 桶的前缀。

前缀必须符合 Amazon S3 密钥命名规则和限制。有关更多信息，请参阅 Amazon S3 开发人员指南中的 对象键和元数据。

有效值 `Prefix` 示例如下所示：

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

类型：字符串

长度约束：最小长度为 0。长度上限为 128。

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ValueHolder

服务：Amazon QLDB

一种可以包含多种编码格式的值的结构。

内容

IonText

`ValueHolder` 结构中包含的 Amazon Ion 明文值。

类型：字符串

长度限制：长度下限为 1。最大长度为 1048576。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Amazon QLDB 会话

Amazon QLDB 会话支持以下数据类型：

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult

- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

服务：Amazon QLDB Session

包含已中止事务的详细信息。

内容

此例外结构的成员取决于上下文。

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

AbortTransactionResult

服务：Amazon QLDB Session

包含已中止事务的详细信息。

内容

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

CommitTransactionRequest

服务：Amazon QLDB Session

包含已提交事务的详细信息。

内容

CommitDigest

指定要提交的事务的提交摘要。对于每个活动事务，都必须传输提交摘要。如果在客户端计算摘要与 QLDB 计算的摘要不匹配，QLDB 会验证`CommitDigest`并拒绝提交，但会出现错误。

该 `CommitDigest` 参数旨在确保 QLDB 在且仅当服务器处理了客户端发送的确切语句集时，才会提交事务，其顺序与客户端发送这些语句的顺序相同，并且没有重复语句。

类型：Base64 编码的二进制数据对象

必需：是

TransactionId

指定要提交事务的事务 ID。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

CommitTransactionResult

服务：Amazon QLDB Session

包含已提交事务的详细信息。

内容

CommitDigest

已提交事务的提交摘要。

类型：Base64 编码的二进制数据对象

必需：否

ConsumedIOs

包含有关已使用的 I/O 请求数量的指标。

类型：IOUsage 对象

必需：否

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

TransactionId

已提交事务的事务 ID。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

EndSessionRequest

服务：Amazon QLDB Session

指定结束会话请求。

内容

此例外结构的成员取决于上下文。

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

EndSessionResult

服务：Amazon QLDB Session

包含已结束会话的详细信息。

内容

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ExecuteStatementRequest

服务：Amazon QLDB Session

指定执行语句的请求。

内容

Statement

指定请求的语句。

类型：字符串

长度限制：长度下限为 1。最大长度为 100000。

必需：是

TransactionId

指定请求的事务 ID。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：是

Parameters

指定请求中参数化语句的参数。

类型：ValueHolder 对象数组

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS 适用于 Ruby V3 的 SDK

ExecuteStatementResult

服务：Amazon QLDB Session

包含已执行语句的详细信息。

内容

ConsumedIOs

包含有关已使用的 I/O 请求数量的指标。

类型：IOUsage 对象

必需：否

FirstPage

包含第一个获取的页面的详细信息。

类型：Page 对象

必需：否

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

FetchPageRequest

服务：Amazon QLDB Session

指定要提取的页面的详细信息。

内容

NextPageToken

指定要获取的页的下一页令牌。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：`^[A-Za-z-0-9+/=]+$`

必需：是

TransactionId

指定要获取的页面的事务 ID。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

FetchPageResult

服务：Amazon QLDB Session

包含已获取的页面。

内容

ConsumedIOs

包含有关已使用的 I/O 请求数量的指标。

类型：IOUsage 对象

必需：否

Page

包含已获取页面的详细信息。

类型：Page 对象

必需：否

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

IOUsage

服务：Amazon QLDB Session

包含已调用命令的 I/O 使用率指标。

内容

ReadIOs

命令发出的读取 I/O 请求的数量。

类型：长整型

必需：否

WriteIOs

命令发出的写入 I/O 请求数。

类型：长整型

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Page

服务：Amazon QLDB Session

包含已获取页面的详细信息。

内容

NextPageToken

下一页的令牌。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：`^[A-Za-z-0-9+/=]+$`

必需：否

Values

一种包含多种编码格式的值的结构。

类型：ValueHolder 对象数组

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

StartSessionRequest

服务：Amazon QLDB Session

指定请求开启一个新会话。

内容

LedgerName

用于开启一个新会话的分类账的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 32。

模式：`(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

StartSessionResult

服务：Amazon QLDB Session

包含已启动的会话的详细信息。

内容

SessionToken

已启动会话的会话令牌。这个 SessionToken 对于当前会话期间发出的每个后续命令都是必需的。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：^[A-Za-z-0-9+/=]+$

必需：否

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

StartTransactionRequest

服务：Amazon QLDB Session

指定启动事务的请求。

内容

此例外结构的成员取决于上下文。

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

StartTransactionResult

服务：Amazon QLDB Session

包含已启动事务的详细信息。

内容

TimingInformation

包含命令的服务器端性能信息。

类型：TimingInformation 对象

必需：否

TransactionId

已启动事务的事务 ID。

类型：字符串

长度限制：固定长度为 22。

模式：`^[A-Za-z-0-9]+$`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

TimingInformation

服务：Amazon QLDB Session

包含命令的服务器端性能信息。Amazon QLDB 捕获从收到请求到发送相应响应之间的时间信息。

内容

ProcessingTimeMilliseconds

QLDB 在处理命令上花费的时间，以毫秒为单位。

类型：长整型

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ValueHolder

服务：Amazon QLDB Session

一种可以包含多种编码格式的值的结构。

内容

IonBinary

`ValueHolder` 结构中包含的 Amazon Ion 二进制值。

类型：Base64 编码的二进制数据对象

长度限制：长度下限为 1。最大长度为 131072。

必需：否

IonText

`ValueHolder` 结构中包含的 Amazon Ion 明文值。

类型：字符串

长度限制：长度下限为 1。最大长度为 1048576。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

常见错误

本部分列出了所有 AWS 服务的常见 API 操作错误。对于特定于此服务的 API 操作的错误，请参阅该 API 操作的主题。

AccessDeniedException

您没有足够的访问权限，无法执行该操作。

HTTP 状态代码：400

IncompleteSignature

请求签名不符合 AWS 标准。

HTTP 状态代码：400

InternalFailure

由于未知错误、异常或故障，请求处理失败。

HTTP 状态代码：500

InvalidAction

所请求的操作无效。验证操作是否已正确键入。

HTTP 状态代码：400

InvalidClientTokenId

在我们的记录中没有所提供的 X.509 证书或 AWS 访问密钥 ID。

HTTP 状态代码：403

NotAuthorized

您无权执行此操作。

HTTP 状态代码：400

OptInRequired

AWS 访问密钥 ID 需要订阅服务。

HTTP 状态代码：403

RequestExpired

请求到达服务的时间超过请求上的日期戳或请求到期日期 (如针对预签名 URL) 15 分钟，或者请求上的日期戳离到期还有 15 分钟以上。

HTTP 状态代码：400

ServiceUnavailable

由于服务器发生临时故障而导致请求失败。

HTTP 状态代码：503

ThrottlingException

由于请求限制而导致请求被拒绝。

HTTP 状态代码：400

ValidationError

输入未能满足 AWS 服务指定的约束。

HTTP 状态代码：400

常见参数

以下列表包含所有操作用于使用查询字符串对 Signature Version 4 请求进行签名的参数。任何特定于操作的参数都列在该操作的主题中。有关 Signature Version 4 的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

Action

要执行的操作。

类型：字符串

必需：是

Version

编写请求所针对的 API 版本，格式为 YYYY-MM-DD。

类型：字符串

必需：是

X-Amz-Algorithm

您用于创建请求签名的哈希算法。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

有效值：`AWS4-HMAC-SHA256`

必需：条件

X-Amz-Credential

凭证范围值，该值是一个字符串，其中包含您的访问密钥、日期、您要定位的区域、您请求的服务以及终止字符串（“aws4_request”）。值采用以下格式表示：access_key/YYYYMMDD/region/service/aws4_request。

有关更多信息，请参阅《IAM 用户指南》中的创建已签名的 AWS API 请求。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

X-Amz-Date

用于创建签名的日期。格式必须为 ISO 8601 基本格式 (YYYYMMDD'T'HHMMSS'Z')。例如，以下日期时间是有效的 X-Amz-Date 值：`20120325T120000Z`。

条件：X-Amz-Date 对于所有请求都是可选的；它可以用于覆盖对请求签名所使用的日期。如果以 ISO 8601 基本格式指定 Date 标头，则不需要 X-Amz-Date。使用 X-Amz-Date 时，它始终会覆盖 Date 标头的值。有关更多信息，请参阅《IAM 用户指南》中的 AWS API 请求签名的元素。

类型：字符串

必需：条件

X-Amz-Security-Token

通过调用 AWS Security Token Service（AWS STS）获得的临时安全令牌。有关支持来自 AWS STS 的临时安全凭证的服务列表，请参阅《IAM 用户指南》中的使用 IAM 的 AWS 服务。

条件：如果您使用来自 AWS STS 的临时安全凭证，则必须包含安全令牌。

类型：字符串

必需：条件

X-Amz-Signature

指定从要签名的字符串和派生的签名密钥计算的十六进制编码签名。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

X-Amz-SignedHeaders

指定作为规范请求的一部分包含的所有 HTTP 标头。有关指定已签名标头的更多信息，请参阅《IAM 用户指南》中的创建已签名的 AWS API 请求。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

Amazon QLDB 资源中的限额和限制

本节介绍 Amazon QLDB 中的当前限额（以前称为限制）。

主题

默认限额

QLDB 具有以下默认限额，AWS 一般参考中的 Amazon QLDB 端点和限额 也列出了这些限额。这些限额按区域分配到每个 AWS 账户。要请求增加区域中的账户限额，请使用服务限额控制台。

登录到 AWS Management Console 并在以下位置打开服务限额控制台：https://console.aws.amazon.com/servicequotas/。

资源	默认限额
您可以在当前区域中的此账户中创建的活动分类账的最大数量	5
每个分类账的活跃日记账导出至 Amazon S3 的最大数量	2
每个分类账流向 Kinesis Data Streams 的最大活跃日记账流数量	5

固定限额

除了默认限额外，QLDB 还为每个分类账设置了以下固定限额。这些限额无法通过服务限额提高：

资源	固定限额
并发活动会话数量	1500
活动表数量	20
总表数量（活动和非活动）	40
每个表格的索引数量	5
事务中的文件数量	40
事务中要编辑的修订次数	1
文档大小 (按IonBinary 格式编码)	128 KB
语句参数大小 (IonBinary 格式)	128 KB
语句参数大小 (IonText格式)	1 MB
语句字符串长度	100000 个字符
事务大小	4MB
事务超时	30 秒
已完成日记账导出作业的到期时间	7 天
终端日记账流到期时间	7 天

> **Note**
>
> 在 QLDB 中，删除表被视为非活动表并计入此总限额。

分类账限额

要请求增加区域中的账户限额，请使用服务限额控制台。

访问 https://console.aws.amazon.com/servicequotas/，打开 Service Quotas 控制台。

部分 QLDB 用例需要根据业务增长情况，增加AWS 账户每个地区的分类账数量。例如，您可能需要创建专用分类账，以隔离客户或数据。在这种情况下，可考虑利用多账户架构使用 QLDB 限额。有关更多信息，请参阅AWS 白皮书 SaaS 租赁个例策略中的账户 Silo 隔离。

文档大小

以IonBinary格式编码的文档的最大大小为 128 KB。我们无法提供 IonText 格式文档大小的精确限制，因为从文本到二进制的转换会因每个文档的结构而有很大差异。QLDB 支持包含开放内容的文档，因此每个独特的文档结构都可改变大小计算方式。

事务大小

QLDB 中的最大事务大小为 4 MB。事务规模是根据以下因素的总和计算得出的。

Deltas

由事务中的所有对语句生成的文档更改。在影响多个文档的事务中，总增量大小就是每个受影响文档的单个增量之和。

元数据

系统生成的、与每个受影响文档关联的事务元数据。

索引

如果在受事务影响的表上定义索引，则关联的索引条目还会生成增量。

历史记录

由于所有文档修订都保留在 QLDB，因此所有事务也会追加到历史记录中。

插入 – 表格中的每个文档也在其历史表中插入了一个副本。例如，新插入的 100 KB 文档在事务中生成至少 200 KB 增量。（这是一个粗略的估计，不包括元数据或索引。）

更新 - 任何文档更新，即使是单个字段，都会在历史记录中创建整个文档的新修订版本，加上或减去更新的增量。这意味着在大型文档中进行少量更新仍会生成较大事务增量。例如，在现有的 100 KB 文档中添加 2 KB 数据会在历史记录中创建新 102 KB 修订版。这加起来至少占事务总增量的 104 KB。（同样，此估计值不包含元数据或索引。）

删除 - 与更新类似的是，任何删除事务都会在历史记录中创建新的文档修订版。但是，新创建的 DELETE 修订版本比原始文档小，因为其用户数据为空且仅包含元数据。

命名约束

下表介绍 Amazon QLDB 中的命名约束。

分类账 日记账流名称	• 必须包含 1–32 个字母数字字符。 • 第一个和最后一个字符必须包含一个字母或数字。 • 不能是全部数字。 • 不能包含两个连续连字符。 • 区分大小写。
表名称	• 标签只能包含 1-128 字母数字字符或下划线字符。 • 第一个字符必须是字母或下划线。 • 其余字符可以是字母数字字符和下划线的任意组合。 • 区分大小写。 • 不能是 QLDB PartiQL 的 保留字。

Amazon QLDB 相关信息

下列相关资源在您使用此服务的过程中会有所帮助。

主题

- 技术文档
- GitHub 存储库
- AWS 博文和文章
- 媒体
- 一般AWS资源

技术文档

- Amazon QLDB 常见问题解答 — 有关产品的常见问题。
- Amazon QLDB 定价 — AWS定价信息和示例。
- AWS re:Post – AWS 提问和答案的社区论坛（问答）。
- Amazon Ion — 开发人员指南、用户指南和 Amazon Ion 数据格式参考资料。
- PartiQL – 查询语言的规范文档和一般教程。
- QLDB 研讨会 — 提供使用 Amazon QLDB 构建记录系统应用程序的实际动手示例的研讨会，包括以下实验：
  - AQLDB 基础知识
  - 使用 Amazon Ion 将 Ion 与 JSON 相互转换 (Java)
  - 使用 AWS Glue 和 Amazon Athena 为数据湖启用 QLDB 数据
  - QLDB 数据流式传输至 Amazon Aurora MySQL DB实例
- 使用 Amazon QLDB 防篡改质量数据 — AWS一种解决方案实施，介绍如何使用 QLDB 维护准确的数据更改历史来防止攻击者篡改质量数据。 AWS解决方案实施可帮您解决常见问题并使用AWS更快地构建。

GitHub 存储库

驱动程序

- .NET 驱动程序 - QLDB 驱动程序的 .NET 实现。

- Go 驱动程序 - QLDB 驱动程序的 Go 实现。
- Java 驱动程序 - QLDB 驱动程序的 Java 实现。
- Node.js 驱动程序 - QLDB 驱动程序的 Node.js 实现。
- Python 驱动程序 - QLDB 驱动程序的 Python 实现。

命令行 Shell

- QLDB Shell – QLDB 事务数据 API 命令行接口的 Python 和 Rust 实现。

示例应用程序

- Java DMV 应用程序 — 基于机动车辆部 (DMV) 用例的教程应用程序。它演示了使用 QLDB 和 Java 版 QLDB 驱动程序的基本操作和最佳实践标准。
- .NET DMV 应用程序 — 基于 DMV 的教程应用程序，演示使用 QLDB 和适用于 .NET 的 QLDB 驱动程序的基本操作和最佳实践。
- Node.js DMV 应用程序 — 基于 DMV 的教程应用程序，演示使用 QLDB 和 Node.js 的 QLDB 驱动程序的基本操作和最佳实践。
- Python DMV 应用程序 — 基于 DMV 的教程应用程序，演示了使用 QLDB 和 Python 的 QLDB 驱动程序的基本操作和最佳实践。
- 分类账加载器 — Java 框架，用于使用支持的交付渠道 (AWS DMS, Amazon SQS、Amazon SNS、Kinesis Data Streams、Amazon MSK 或 EventBridge) 将数据高速异步加载到 QLDB 分类账。
- 导出处理器 — 可扩展的 Java 框架，通过读取导出的输出并按顺序遍历导出的块，处理在 Amazon S3 中处理 QLDB 导出的工作。
- QLDB 在 Python 中流式传输示例 Lambda — 该应用程序演示如何 AWS Lambda 使用函数将 QLDB 数据发送到订阅了Amazon SQS 队列的Amazon SNS 主题，从而使用 QLDB 流。
- QLDB 直播 OpenSearch 集成示例 — Python 应用程序，演示如何使用流将Amazon OpenSearch Service 服务与 QLDB 集成。
- 双条目应用程序 – Java 应用程序，演示如何使用 QLDB 对复式记账财务分类账应用程序进行建模。
- QLDB KVS for Node.js – QLDB 的简单键值存储接口库，具有用于文档验证的额外功能。

Amazon Ion 和 PartiQL

- Amazon Ion 库 — Ion 团队支持的库、工具与文档。

- PartiQL 实现 – PartiQL 的实现、规范和教程。

AWS 博文和文章

- Earnin 如何使用Amazon QLDB 构建分类账服务（2023 年 2 月 16 日）— 介绍 Earnin.com 如何使用 QLDB 构建分类账服务，为用户提供功能齐全的现代移动金融应用程序。
- 使用 AWS Glue 和 Amazon Athena 导出和分析 Amazon QLDB 日志数据（2022 年 12 月 19 日）— 讨论如何使用 QLDB 中的导出功能和AWS Glue，为基于分类账的架构提供报告和分析功能。
- Shinsegae International 如何利用 Amazon QLDB 增强客户体验并防止假冒 (2022 年 8 月 3 日) — 描述 Shinsegae International 如何使用Amazon QL DB 构建数字真伪验证服务，告知客户奢侈品的真实性，并提供产品的购买和分销历史记录。
- BungkusIT 使用 Amazon QLDB 和 VeriDoc Global 的 ISV 技术来改善客户和送货代理体验 (2022 年 4 月 29 日) – 介绍一家名为 BungkuSit 的物流公司如何使用QLDB实施一个集中、透明的平台，用于通过不可变的、可加密验证的交易日志进行跨行业通信。
- 使用 Amazon QLDB 作为带有 REST API 和 JSON 的不可变键值存储(2022 年 2 月 14 日)— 引入了一种将 QLDB 用作不可变键值存储并通过 REST API 使用其审计功能的方法。
- 使用 Amazon QLDB 构建核心银行系统 (2022 年 1 月 21 日)— 展示如何使用 QLDB 构建核心银行分类账系统。它还说明了为什么 QLDB 是一个适合金融服务行业需求的理想数据库。
- Specright 如何使用 Amazon QLDB 创建可追踪的供应链网络（2022年1月21日） — 介绍 Specright 如何使用 QLDB 创建可追溯的供应链网络，使批发品牌、零售商和制造商能够共享关键的供应链数据和包装规格数据。
- fEMR 如何使用 Amazon QLDB 提供加密安全且可验证的医疗数据(2021 年 12 月 23 日)——探索 FemR 团队如何使用 QLDB 和其他AWS托管服务来支持他们的救济工作。
- 监控 Amazon QLDB 查询访问模式（2021 年 11 月 8 日） — 介绍如何将 QLDB 交易和交易中运行的 PartiQL 语句与通过 Amazon API Gateway 收到的 API 请求关联起来。
- 使用 AWS Lambda在 Amazon QLDB 构建简单的 CRUD 操作和数据流 (2021 年 9 月 28 日) – 演示如何使用AWS Lambda函数对 QLDB 执行 CRUD（创建、读取、更新和删除）操作。
- 使用 Amazon QLDB 进行真实世界的加密验证 (2021 年 8 月 26 日)— 在现实用例的背景下，讨论 QLDB 分类账中加密验证的价值。
- 使用 Accord Project 和 Amazon QLDB 验证交付条件：第 1 部分 、第 2 部分（2021 年 6 月 28 日）— 讨论如何应用智能法律合约技术通过开源 Accord 项目 和 QLDB 验证交付条件。

- Amazon QLDB 数据流通过AWS CDK (2021 年 6 月 7 日) – 展示如何使用 AWS Cloud Development Kit (AWS CDK) 来设置 QLDB，以及使用AWS Lambda使用函数填充 QLDB 数据，以及如何设置 QLDB 流以提供分类账数据的数据弹性。
- 演示 QLDB 中的精细访问控制 (2021 年 6 月 1 日) — 介绍如何开始使用 QLDB 分类账的精细AWS Identity and Access Management (IAM) 权限。
- 使用 DynamoDB 流和 QLDB 流进行流处理 (2020 年 11 月 23 日) – 提供 DynamoDB 流和 QLDB 流之间的简要比较，以及有关它们的入门技巧。
- 将数据从 Amazon QLDB 流式传输到 OpenSearch (2020 年 8 月 19 日) – 描述如何将数据从 QLDB 流式传输到 Amazon OpenSearch Service 以支持富文本搜索和下游分析，例如跨记录的聚合或指标。
- 我如何通过 Nodejs 近乎实时地将数据从 Amazon QLDB 流式传输到 DynamoDB(2020 年 7 月 7 日)— 介绍如何将数据从 QLDB 流式传输至 DynamoDB 以支持个位数的延迟和可无限扩展的键值查询。
- 通过 AWS AppSync构建 Amazon QLDB 的 GraphQL 接口：第 1 部分、第 2 部分 (2020 年 5 月 4 日) – 介绍如何集成 QLDB 和 AWS AppSync，QLDB 分类账之上提供多功能、由 GraphQL 支持的 API。

媒体

AWS 视频

- AWS教程和演示：QLDB 至 Aurora 流式传输 （2023 年 3 月 17 日；21分钟 ）— 该视频解释了实现 QLDB 流式传输功能的基本概念，并演示了如何设置从 QLDB 到 Amazon Aurora MySQL 下游数据库的数据流。
- ArcBlock：利用Amazon QLDB 构建去中心化身份解决方案（2022 年 5 月 31 日；4 分钟）— ArcBlock 介绍了他们使用 QLDB 构建的去中心化身份解决方案。
- AWSre: Invent 2020：使用 Amazon QLDB 作为核心业务应用程序的信任系统数据库（2021 年 2 月 5 日；30 分钟）— 了解 Amazon QLDB 用户如何早期将分类账数据库的独特属性应用于数据来源和加密可验证性来实施内置数据完整性的记录系统。
- AWSre: Invent 2020：使用Amazon QLDB 构建无服务器应用程序（2021 年 2 月 5 日；28 分钟）— 了解如何通过将Amazon QLDB 与Amazon Kinesis 和 Amazon DynamoDB 等服务相结合，构建、测试和优化功能齐全的无服务器应用程序。AWS Lambda
- AWSre: Invent 2020：使用 Amazon QLDB 构建可保持数据完整性的基于审计的应用程序(2021 年 2 月 5 日；18 分钟) — 本课程深入探讨了 Amazon QLDB 可以解决的问题，回答您关于何时以及为何使用分类账数据库的问题，并分享了 Osano 等客户的用例。

- 如何使用 Amazon QLDB 和 .NET 应用程序集中存储不可变日志 (2020 年 12 月 7 日；10分钟) –演示如何将 QLDB 与 .NET 应用程序结合使用来集中存储不可变日志。
- 虚拟研讨会：使用 QLDB 和 Java 构建基于分类账的记录系统 - AWS 在线技术讲座 (2020 年 7 月 29 日； 87分钟) – 由讲师指导的研讨会，介绍使用 Ion Immersion Day 实验室，以使用 Amazon Ion 库和适用于 Java 的 QLDB 驱动程序构建数据加载程序。
- 开发者研讨会：在 ABT 节点上使用AWS不可变分类账数据库 QLDB (2020 年 6 月 22 日；34分钟) – 有关如何在 ABT 节点上设置和配置 QLDB 的分步指南。它还介绍了如何安装和配置 ArcBlock 的区块链浏览器和 Boarding Gate Blocklet 以连接质 QLDB。
- AWS 技术讲座：客户对使用 Amazon QLDB 构建事件触发的记录系统应用程序的看法 (2020 年 3 月 31 日；29 分钟) — 英国驾驶员和车辆许可机构 (DVLA) 的AWS数据英雄兼首席架构师 Matt Lewis 的技术演讲解释了 DVLA 的 QLDB 用例及其应用程序的事件驱动架构。
- AWSre: Invent 2019：为什么需要分类账数据库：BMW、DVLA和Sage讨论用例（2019 年 12 月 5 日；47 分钟） — 客户 BMW、英国政府组织 DVLA 和 Sage 的演示，讨论了使用账本数据库的原因并分享了他们的 QLDB 用例。
- AWSre: Invent 2019：Amazon QLDB：一位工程师深入探讨它为何能改变游戏规则（2019 年 12 月 5 日；50 分钟） — Andrew certain（AWS杰出工程师）的演讲讨论了 QLDB 独特的、期刊优先的架构及其各种创新。它包含加密哈希、默克尔树、多可用区复制和 PartiQL 支持。
- 使用首创的分类账数据库 Amazon QLDB 构建应用程序 — AWS在线技术讲座（2019 年 11 月 19 日；51 分钟）——场深入的技术讲座，描述了 QLDB 的独特功能以及如何使用核心功能的细节。它包括查询数据的完整历史记录、对文档加密验证以及设计数据模型。

播客

- 买家为何选择 Amazon QLDB？（2020 年 7 月 5 日；33 分钟） — 本次讨论解释了分类账数据库的定义、分类账数据库与区块链的不同之处以及当今客户如何使用分类账数据库。

一般AWS资源

- 课程和研讨会 – 指向基于角色的专业课程和自主进度动手实验室的链接，这些课程和实验室旨在帮助您增强 AWS 技能并获得实践经验。
- AWS 开发人员中心 – 浏览教程、下载工具并了解 AWS 开发人员活动。
- AWS 开发人员工具 – 指向开发人员工具、开发工具包、IDE 工具包和命令行工具的链接，这些资源用于开发和管理 AWS 应用程序。
- 入门资源中心 – 了解如何设置 AWS 账户、加入 AWS 社区和启动您的第一个应用程序。

- 动手实践教程 – 按照分步教程在 AWS 上启动您的第一个应用程序。
- AWS 白皮书 – 指向 AWS 技术白皮书的完整列表的链接，这些资料涵盖了架构、安全性、经济性等主题，由 AWS 解决方案架构师或其他技术专家编写。
- AWS Support 中心 – 用于创建和管理 AWS Support 案例的中心。还提供指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况以及AWS Trusted Advisor。
- AWS Support – 提供有关 AWS Support 的信息的主要网页，这是一个一对一的快速响应支持渠道，可以帮助您在云中构建和运行应用程序。
- 联系我们 – 用于查询有关AWS账单、账户、事件、滥用和其他问题的中央联系点。
- AWS 网站条款 – 有关我们的版权和商标、您的账户、许可、网站访问和其他主题的详细信息。

Amazon QLDB 发布历史记录

下表描述 Amazon QLDB 每次发布时进行的重要修改和 Amazon QLDB 开发人员指南 中的相应更新。如需对此文档更新的通知，您可以订阅 RSS 源。

- API 版本： 2019-01-02
- 文档最新更新时间：2023 年 1 月 3 日

变更	说明	日期
更新 IAM 指南	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践。	2023 年 1 月 3 日
对 AWS 托管式策略的更新	Amazon QLDB 更新了AWS现有的托管策略`AmazonQLDBFullAccess` 和`AmazonQLDBConsoleFullAccess` 。这些策略具有新权限，允许主题使用 PartiQL 存储过程编校文档修订版。有关更多信息，请参阅 Amazon QLDB 的 AWS 托管式策略。	2022 年 11 月 4 日
数据编校	Amazon QLDB 现在支持 2021 年 7 月 22 日当天或之后创建的分类账的 `REDACT_REVISION` PartiQL 存储过程。使用此存储过程，您可永久删除历史记录中处于非活动状态的文档修订版，同时仍能保持分类账的整体数据完整性。有关更多信息，请参阅 编校文档修订版。	2022 年 11 月 3 日

Node.js 驱动程序 v3	适用于 Node.js 版本 3.0 的 Amazon QLDB 驱动程序现已正式上线。此版本引入了对 AWS SDK for JavaScript v3 的支持。有关发行说明，请参阅 GitHub 存储库awslabs/amazon-qldb-driver-nodejs。	2022 年 9 月 26 日
Go 驱动程序 v3	适用于 Go 版本 3.0 的 Amazon QLDB 驱动程序现已正式上线。此版本引入了对 AWS SDK for Go v2 的支持。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-go。	2022 年 8 月 11 日
全新 PartiQL 查询编辑器	Amazon QLDB 控制台推出了新的 PartiQL 查询编辑器。新的 QLDB PartiQL 编辑器为编写查询、调试事务和探索结果提供了改进后的界面。有关打开和使用编辑器的信息，请参阅使用控制台访问 Amazon QLDB。	2022 年 6 月 22 日
.NET 驱动程序的 Ion 对象映射器	适用于 .NET version 1.3 的 Amazon QLDB 驱动程序引入了对 Ion 对象映射器的支持。此功能让您完全无需在 Amazon Ion 类型和原生 C# 类型之间手动转换。有关 Ion 对象映射器的完整更改历史记录，请参阅 GitHub 存储库`amzn/ion-object-mapper-dotnet` 中的 CHANGELOG.md 文件。	2022 年 1 月 19 日

JSON 日记账导出格式	Amazon QLDB 现在支持日记账导出的JSON 行 输出格式。有关更多信息，请参阅将日记账数据从 Amazon QLDB 导出。	2021 年 12 月 21 日
Amazon QLDB 故障排除	添加了新的疑难解答主题，该主题为使用 Amazon QLDB 时可能遇到的常见错误的汇总列表提供了指导。	2021 年 12 月 8 日
新上线区域	Amazon QLDB 现已在加拿大（中部）区域中提供。有关可用区域的完整列表，请参阅 Amazon Web Services 一般参考 中的 Amazon QLDB 端点和限额。	2021 年 11 月 11 日
防止跨服务混淆代理	Amazon OpenSearch Service 支持在 IAM 资源策略中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键，以防止出现混淆代理问题。有关更多信息，请参阅防止跨服务混淆代理。	2021 年 11 月 8 日
Amazon QLDB Shell v2	使用 Rust 编写的 Amazon QLDB Shell 2.0 版现已正式上线。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-shell。	2021 年 10 月 14 日

对 AWS 托管式策略的更新	Amazon QLDB 更新了AWS现有的托管策略`AmazonQLDBFullAccess` 和`AmazonQLDBConsoleFullAccess` 。这些策略新增了权限，允许委托人将您账户中的任何 IAM 角色资源传递至 QLDB 服务。这是所有日记账流请求所必需的。有关更多信息，请参阅 Amazon QLDB 的 AWS 托管式策略。	2021 年 9 月 2 日
客户托管 AWS KMS 密钥	Amazon QLDB 现在支持使用 AWS Key Management Service (AWS KMS) 中的客户托管密钥对新的分类账资源进行静态加密。有关更多信息，请参阅 Amazon QLDB 中的静态加密。	2021 年 7 月 22 日
更新 AWS 托管式策略	Amazon QLDB 更新了现有AWS托管策略`AmazonQLDBReadOnly` ，删除了之前列出两次的重复`qldb:GetBlock` 操作。有关更多信息，请参阅 Amazon QLDB 的 AWS 托管式策略。	2021 年 7 月 1 日
新上线区域	Amazon QLDB 现已在欧洲（伦敦）区域中提供。有关可用区域的完整列表，请参阅 Amazon Web Services 一般参考 中的 Amazon QLDB 端点和限额。	2021 年 6 月 24 日

对 AWS 托管式策略的更新	Amazon QLDB 更新了AWS现有的托管策略`AmazonQLDBFullAccess` 和`AmazonQLDBConsoleFullAccess` 。这些策略新增了权限，允许主体更新所有分类账中的权限模式，并在所有STANDARD权限分类账中运行所有 PartiQL 命令。有关更多信息，请参阅 Amazon QLDB 的 AWS 托管式策略。	2021 年 5 月 27 日
标准权限模式	Amazon QLDB 现在支持分类账资源STANDARD权限模式。在标准权限模式下，可以对分类账、表格和 PartiQL 命令进行更精细粒度的访问控制。有关信息，请参阅 标准权限模式入门。	2021 年 5 月 27 日
PartiQL 语句统计信息	PartiQL 语句统计信息功能现已在 Amazon QLDB 控制台查询编辑器中提供。有关更多信息，请参阅 获取语句统计信息。	2021 年 5 月 24 日
教程：使用 AWS 软件开发工具包验证数据	添加了分步教程，其中包含代码示例，演示了如何通过AWS软件开发工具包使用 QLDB API 在 Amazon QLDB 中验证修订哈希和区块哈希。要了解更多信息，请参阅教程：使用 AWSSDK 验证数据。	2021 年 5 月 6 日

.NET 驱动程序 v1.2	适用于.NET 版本 1.2 的 Amazon QLDB 驱动程序现已正式上线。此版本引入异步 API。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-dotnet。	2021 年 4 月 1 日
PartiQL DROP INDEX语句	Amazon QLDB 中的 PartiQL 现支持 删除索引语句。有关更多信息，请参阅 删除索引。	2021 年 3 月 3 日
PartiQL 语句统计信息	PartiQL 语句统计功能现已在最新版本的 Amazon QLDB 驱动程序中提供，适用于所有支持的语言，包括 .NET、Go 和 Python。有关更多信息，请参阅 获取语句统计信息。	2021 年 2 月 25 日
TypeScript 快速入门教程	在 Node.js Amazon QLDB 驱动程序的快速入门教程中添加 TypeScript 代码示例。	2020 年 12 月 28 日
PartiQL 语句统计信息	最新版本的 Java 和 Node.js 版 Amazon QLDB 驱动程序现在提供语句执行统计信息，可以帮助您更高效运行 PartiQL 语句。有关更多信息，请参阅 获取语句统计信息。	2020 年 12 月 22 日
驱动程序 cookbook 参考和快速入门教程	添加了 Go 和 Node.js 驱动程序 cookbook 参考、Java 和 Python 驱动程序的快速入门教程以及在 QLDB 中使用 Amazon Ion 的指南。有关更多信息，请参阅 Amazon QLDB 驱动程序入门。	2020 年 11 月 24 日

Amazon QLDB Shell v1.1	Amazon QLDB Shell 1.1 版本现已正式上线。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-shell。	2020 年 10 月 26 日
适用于 Go 的 Amazon QLDB 驱动程序	适用于 Go 的 Amazon QLDB 驱动程序 现已正式上线。此驱动程序在 GitHub 上开源，允许您使用AWS SDK for Go 与 QLDB 的事务数据 API 进行交互。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-go。	2020 年 10 月 20 日
Node.js 驱动程序安装建议	添加了新章节，提供了适用于 Node.js 的 Amazon QLDB 驱动程序安装建议，包括如何通过使用保持活动状态重复使用连接来减少延迟。	2020 年 10 月 16 日
在非空表创建索引	Amazon QLDB 现支持在非空表上创建索引。有关更多信息，请参阅管理索引。	2020 年 9 月 30 日
优化查询性能	添加了新章节，描述了 Amazon QLDB 中的查询限制，并提供了在这些限制条件下优化查询性能指导。	2020 年 9 月 18 日
Java 驱动程序的 Cookbook 参考	添加 Cookbook 参考，其中显示了 Java 版 Amazon QLDB 驱动程序常见用例的代码示例。	2020 年 9 月 3 日

驱动程序会话管理	添加了新部分，概述了在 Amazon QLDB 中使用驱动程序进行会话管理，并描述了运行数据事务时 QLDB 驱动程序如何处理会话。	2020 年 9 月 1 日
Java 驱动程序 v2.0	适用于 Java 版本 2.0 的 Amazon QLDB 驱动程序现已正式上线。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-java。	2020 年 8 月 28 日
Node.js 驱动程序 v2.0	适用于 Node.js 版本 2.0 的 Amazon QLDB 驱动程序现已正式上线。有关发行说明，请参阅 GitHub 存储库awslabs/amazon-qldb-driver-nodejs。	2020 年 8 月 27 日
相关信息	添加了一个新主题，其中包含相关信息链接和其他资源，帮助您了解和使用 Amazon QLDB。	2020 年 8 月 24 日
Python 驱动程序 v3.0	适用于 Python 版本 3.0 的 Amazon QLDB 驱动程序现已正式上线。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-python。	2020 年 8 月 20 日
适用于 Go 的 Amazon QLDB 驱动程序	适用于 Go 的 Amazon QLDB 驱动程序的预览版现已上线。此驱动程序允许您使用AWS SDK for Go与 QLDB 的事务数据 API 进行交互。有关更多信息，请参阅 适用于 Go 的 Amazon QLDB 驱动程序（预览版）。	2020 年 8 月 6 日

Python 驱动程序的 Cookbook 参考	添加了 Cookbook 参考资料，其中显示了 Python 版 Amazon QLDB 驱动程序常见用例代码示例。	2020 年 7 月 24 日
Amazon QLDB 中的唯一编号	添加了描述 Amazon QLDB 中唯一 ID 的属性和使用指南的新章节。	2020 年 7 月 9 日
日记账流 AWS CloudFormation 资源	Amazon QLDB 现在支持使用AWS CloudFormation模板创建日记账流资源。有关更多信息，请参阅 AWS CloudFormation用户指南 中的 AWS::QLDB::流。	2020 年 7 月 9 日
.NET 驱动程序的 Cookbook 参考	添加了 Cookbook 参考资料，其中显示了适用于 .NET 的 Amazon QLDB 驱动程序常见用例代码示例。	2020 年 7 月 1 日
适用于 .NET 的 Amazon QLDB 驱动程序	适用于 .NET 的 Amazon QLDB 驱动程序 现已正式上线。此驱动程序在 GitHub 上开源，允许您使用AWS SDK for .NET与 QLDB 的事务数据 API 进行交互。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-dotnet。	2020 年 6 月 26 日

适用于 Node.js 的 Amazon QLDB 驱动程序	适用于 Node.js 的 Amazon QLDB 驱动程序 现已正式上线。该驱动程序在 GitHub 上开源，允许在 Node.js 中使用适用于 JavaScript 的 AWS SDK 与 QLDB 的事务数据 API 进行交互。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-nodejs。	2020 年 6 月 5 日
Amazon QLDB 日记账流	您可以在 Amazon QLDB 中创建一个 流 ，其捕获提交到您日记账的每个文档修订版本，并将此数据实时传送到 Amazon Kinesis Data Streams。有关更多信息，请参阅 从 Amazon QLDB 流式传输日记账数据。	2020 年 5 月 19 日
Amazon Ion 代码示例	添加使用适用于 Java、Node.js 和 Python 的 QLDB 驱动程序查询和处理 Amazon QLDB 分类账的 Amazon Ion 数据的代码示例。有关更多信息，请参阅 Amazon QLDB 中的 Ion 代码示例。	2020 年 5 月 12 日
并发模型和日记账内容	扩展了 Amazon QLDB 并发模型主题，添加了有关使用索引限制乐观并发控制 (OCC) 冲突的信息。添加了一个新章节，描述Amazon QLDB 中的日记账内容。	2020 年 5 月 4 日
Node.js 驱动程序快速入门指南	为 Node.js 的 Amazon QLDB 驱动程序添加快速入门指南。	2020 年 5 月 1 日

Amazon QLDB Shell	Amazon QLDB Shell现已正式上线。这个 Shell 在 GitHub 上开源，它提供了命令行界面，使您能够对分类账数据执行 PartiQL 语句。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-shell。	2020 年 4 月 20 日
合规性认证	Amazon QLDB 现已通过合规计划认证，包括 AWS HIPAA 和 ISO。有关更多信息，请参阅 Amazon QLDB 的合规验证。	2020 年 4 月 3 日
扩展驱动程序推荐	扩展了 Amazon QLDB 驱动程序建议部分，使其适用于所有支持的编程语言。	2020 年 4 月 2 日
Amazon QLDB Shell	Amazon QLDB Shell 的预览版现已上线。这个 Shell 提供了命令行界面，使您能够对分类账数据执行 PartiQL 语句。有关更多信息，请参阅 使用 QLDB Shell 访问 Amazon QLDB (仅数据 API)（预览版）。	2020 年 3 月 23 日
Java 驱动程序 v1.1	适用于 Java 版本 1.1 的 Amazon QLDB 驱动程序现已正式上线。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-java。	2022 年 3 月 20 日

适用于 .NET 的 Amazon QLDB 驱动程序	Amazon QLDB 现在提供.NET 驱动程序的预览版。此驱动程序允许您使用AWS SDK for .NET与 QLDB 的事务数据 API 进行交互。有关更多信息，请参阅 适用于 .NET 的 Amazon QLDB 驱动程序（预览版）。	2020 年 3 月 13 日
适用于 Python 的 Amazon QLDB 驱动程序	适用于 Python 的 Amazon QLDB 驱动程序 现已正式上线。此驱动程序在 GitHub 上开源，允许您使用AWS SDK for Python (Boto3)与 QLDB 的事务数据 API 进行交互。有关发行说明，请参阅 GitHub 存储库 awslabs/amazon-qldb-driver-python。	2020 年 3 月 11 日
PartiQL UNDROP TABLE 语句和嵌套 DML	Amazon QLDB 中的 PartiQL 现在支持数据操作语言 (DML) 语句，在此语句中，嵌套集合被指定为FROM子句中的源。有关更多信息，请参阅 PartiQL 参考中的FROM 语句。QLDB PartiQL 还支持 UNDROP TABLE 语句。有关更多信息，请参阅 撤销删除表。	2020 年 2 月 26 日
扩展了介绍主题	扩展了入门内容什么是 Amazon QLDB？主题，纳入包括新的 Amazon QLDB 概述和从关系型到分类账。	2020 年 1 月 24 日

支持函数的 PartiQL 参考	扩展了 Amazon QLDB PartiQL 参考，增加了有关支持的 SQL 函数的详细使用信息。有关更多信息，请参阅PartiQL 函数。	2020 年 1 月 2 日
驱动程序建议与常见错误	新增了描述适用于 Java 的 Amazon QLDB 驱动程序的驱动程序建议以及所有驱动程序语言的常见错误的新章节。	2020 年 1 月 2 日
新上线区域	Amazon QLDB 目前在亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）和欧洲地区（法兰克福）区域提供。有关可用区域的完整列表，请参阅 Amazon Web Services 一般参考 中的 Amazon QLDB 端点和限额。	2019 年 11 月 19 日
适用于 Node.js 的 Amazon QLDB 驱动程序	Amazon QLDB 现在提供了 Node.js 驱动程序的预览版。该驱动程序让您能够在 Node.js 中使用适用于 JavaScript 的 AWS SDK 与 QLDB 的事务数据 API 进行交互。有关更多信息，请参阅 适用于 Node.js 的 Amazon QLDB 驱动程序（预览版）。	2019 年 11 月 13 日
适用于 Python 的 Amazon QLDB 驱动程序	Amazon QLDB 现在提供了 Python 驱动程序的预览版。此驱动程序允许您使用AWS SDK for Python (Boto3)与 QLDB 的事务数据 API 进行交互。有关更多信息，请参阅 适用于 Python 的 Amazon QLDB 驱动程序（预览版）。	2019 年 10 月 29 日

公开发布版	这是 Amazon QLDB 的第一个公开发布版。本版本包括开发人员指南和集成的日记账管理 API 参考。	2019 年 9 月 10 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。