aws

Guía para desarrolladores

Amazon Quantum Ledger Database (Amazon QLDB)

Amazon Quantum Ledger Database (Amazon QLDB): Guía para desarrolladores

Table of Contents

¿Qué es Amazon QLDB?

Amazon Quantum Ledger Database (Amazon QLDB) es una base de datos de libro mayor completamente administrada en la que se proporciona un registro de transacciones transparente, inmutable y que se puede verificar mediante criptografía, cuya propiedad denota una autoridad central de confianza. Amazon QLDB se puede usar para hacer un seguimiento de todos los cambios de los datos de una aplicación, así como para mantener un historial de cambios completo y que se pueda verificar con el paso del tiempo. Para obtener más información sobre la variedad de opciones de bases de datos disponibles en Amazon Web Services, consulte Choosing the right database for your organization on AWS (Elegir la base de datos adecuada para su organización en AWS).

Los libros mayores suelen usarse para registrar el historial de la actividad económica y financiera de una organización. Muchas organizaciones crean aplicaciones con una funcionalidad similar a la de un libro mayor porque desean mantener un historial preciso de los datos de sus aplicaciones. Por ejemplo, es posible que deseen supervisar el historial de créditos y débitos en las transacciones bancarias, verificar el linaje de datos de una reclamación de seguros o rastrear el movimiento de un artículo en una red de cadena de suministro. Las aplicaciones de libro mayor suelen implementarse mediante tablas de auditoría personalizadas o registros de auditoría creados en bases de datos relacionales.

Amazon QLDB es una nueva clase de base de datos que acaba con la necesidad de llevar a cabo el complejo esfuerzo de desarrollo que supone crear aplicaciones de libro mayor propias. Con QLDB, el historial de cambios de sus datos es inmutable: no se pueden sobrescribir ni modificar. Además, la criptografía permite comprobar que no se han producido cambios involuntarios en los datos de su aplicación. QLDB emplea un registro transaccional inmutable conocido como diario. El diario es solamente un anexo y está compuesto por un conjunto de bloques secuenciados y encadenados que contienen los datos confirmados.

Vídeo de Amazon QLDB

Para obtener más información sobre Amazon QLDB y sus beneficios, vea este vídeo de información general sobre QLDB en YouTube.

Precios de Amazon QLDB

Con Amazon QLDB, solo paga por lo que usa. No hay cuotas mínimas ni uso obligatorio de servicios. Solo tiene que pagar por los recursos que consuma su base de datos de libro mayor , sin necesidad de aprovisionar por adelantado.

Para obtener más información, consulte Precios de Amazon QLDB.

Introducción a QLDB

Le recomendamos que empiece leyendo los siguientes temas:

- Descripción general de Amazon QLDB: para obtener información general de alto nivel sobre QLDB.
- Conceptos básicos y terminología de Amazon QLDB: para aprender los principales conceptos y terminología de QLDB.
- Acceso a Amazon QLDB: para aprender a acceder a QLDB mediante AWS Management Console, API o AWS Command Line Interface (AWS CLI).
- Cómo funciona Amazon QLDB con IAM: para aprender a controlar el acceso a QLDB mediante AWS Identity and Access Management (IAM).

Para comenzar a utilizar rápidamente la consola de QLDB, consulte Introducción a la consola de Amazon QLDB.

Para obtener más información sobre el desarrollo con QLDB usando un controlador proporcionado por AWS, consulte Introducción al controlador Amazon QLDB.

Descripción general de Amazon QLDB

En las secciones siguientes se incluye una descripción general de alto nivel sobre los componentes del servicio de Amazon QLDB y las interacciones entre ellos.

Temas

- El diario primero
- Inmutable

- Verificable criptográficamente
- Similar a SQL y flexible en cuanto a documentos
- Herramientas de código abierto para desarrolladores
- Sin servidor y con alta disponibilidad
- Nivel empresarial

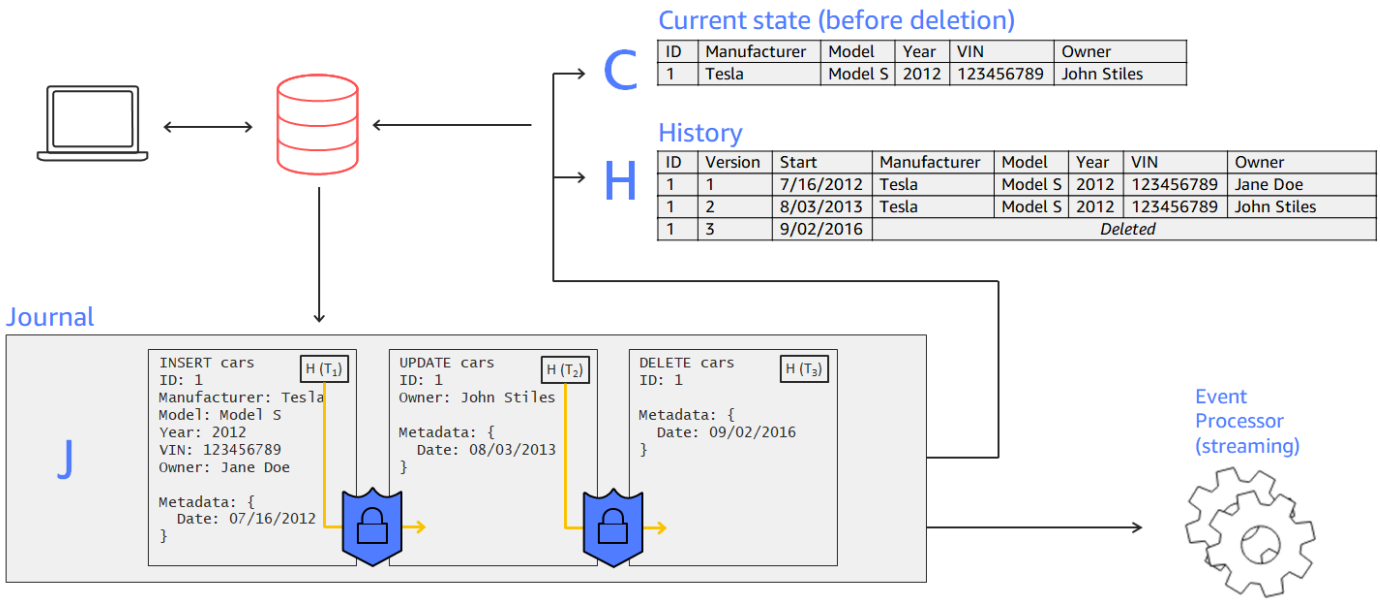
El diario primero

En la arquitectura de bases de datos tradicional, los datos suelen escribirse en tablas como parte de una transacción. Un registro de transacciones (de implementación interna, normalmente) registra todas las transacciones y las modificaciones que estas realizan en la base de datos. El registro de transacciones es un componente fundamental de la base de datos. Necesita este registro para reproducir las transacciones en caso de fallo del sistema, recuperación de desastres o replicación de datos. Sin embargo, los registros de transacciones de las bases de datos no son inmutables, ni están diseñados para proporcionar un acceso fácil y directo a los usuarios.

En Amazon QLDB, el diario es el núcleo de la base de datos. Similar en su estructura a un registro de transacciones, el diario es una estructura de datos inmutable que existe únicamente como anexo y que almacena los datos de la aplicación junto con los metadatos asociados. Todas las transacciones de escritura, incluidas las actualizaciones y las eliminaciones, se confirman primero en el diario.

QLDB emplea el diario para determinar el estado actual de los datos del libro mayor materializándolos en tablas consultables definidas por el usuario. Estas tablas también proporcionan un historial accesible de todos los datos de las transacciones, incluidas las revisiones de los documentos y los metadatos. Además, el diario gestiona la concurrencia, la secuenciación, la verificación criptográfica y la disponibilidad de los datos del libro mayor.

El siguiente diagrama ilustra la arquitectura del diario QLDB.



- En este ejemplo, una aplicación se conecta a un libro mayor y ejecuta transacciones que insertan, actualizan y eliminan un documento en una tabla denominada `cars`.
- Los datos se escriben primero en el diario en orden secuencial.
- A continuación, los datos se materializan en la tabla con vistas integradas. Estas vistas permiten consultar tanto el estado actual como el historial completo del vehículo, y se asigna un número de versión a cada revisión.
- También puede exportar o transmitir datos directamente desde el diario.

Inmutable

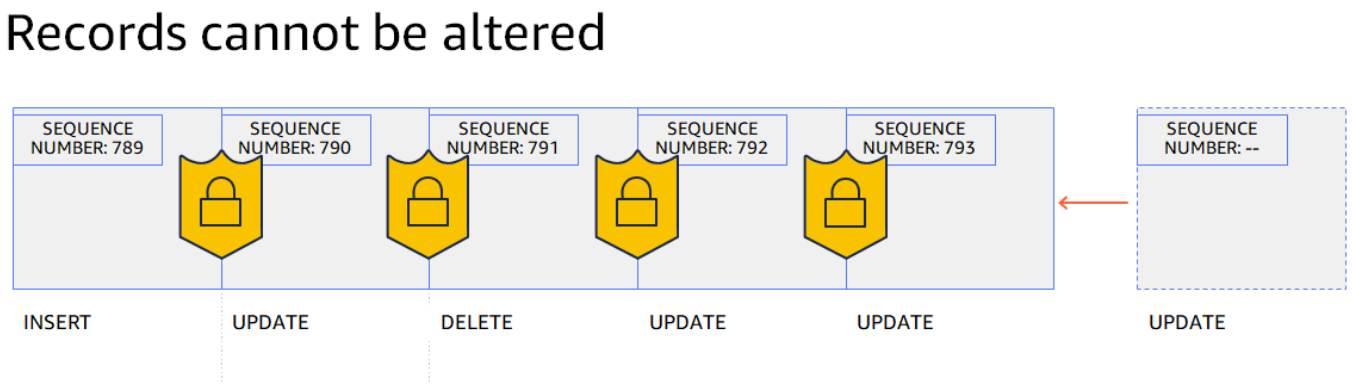
Como el diario de QLDB es únicamente un anexo, mantiene un registro completo de todos los cambios en los datos que no se puede modificar ni sobrescribir. No es posible alterar los datos confirmados mediante API ni ningún otro método. Esta estructura de diario le permite acceder al historial completo de su libro mayor y consultarlo.

> **Note**
>
> La única excepción a la inmutabilidad de QLDB es la redacción de datos. Con esta función, puede cumplir con legislación regulatoria como el Reglamento General de Protección de

Datos (RGPD) de la Unión Europea y la Ley de Privacidad del Consumidor de California (CCPA).
QLDB también permite realizar una operación de edición de datos para eliminar permanentemente las revisiones de documentos inactivos del historial de una tabla. Esta operación elimina solo los datos de usuario de la revisión especificada, sin alterar el diario ni los metadatos del documento. Esto mantiene la integridad general de los datos del libro mayor. Para obtener más información, consulte Editar revisiones de documentos.

QLDB escribe un bloque en el diario de una transacción. Cada bloque contiene objetos de entrada que representan los documentos que inserta, actualiza y elimina, junto con las instrucciones que ejecutó para confirmar dichas acciones. Estos bloques están secuenciados y encadenados para garantizar la integridad de los datos.

El siguiente diagrama ilustra la estructura de este diario.

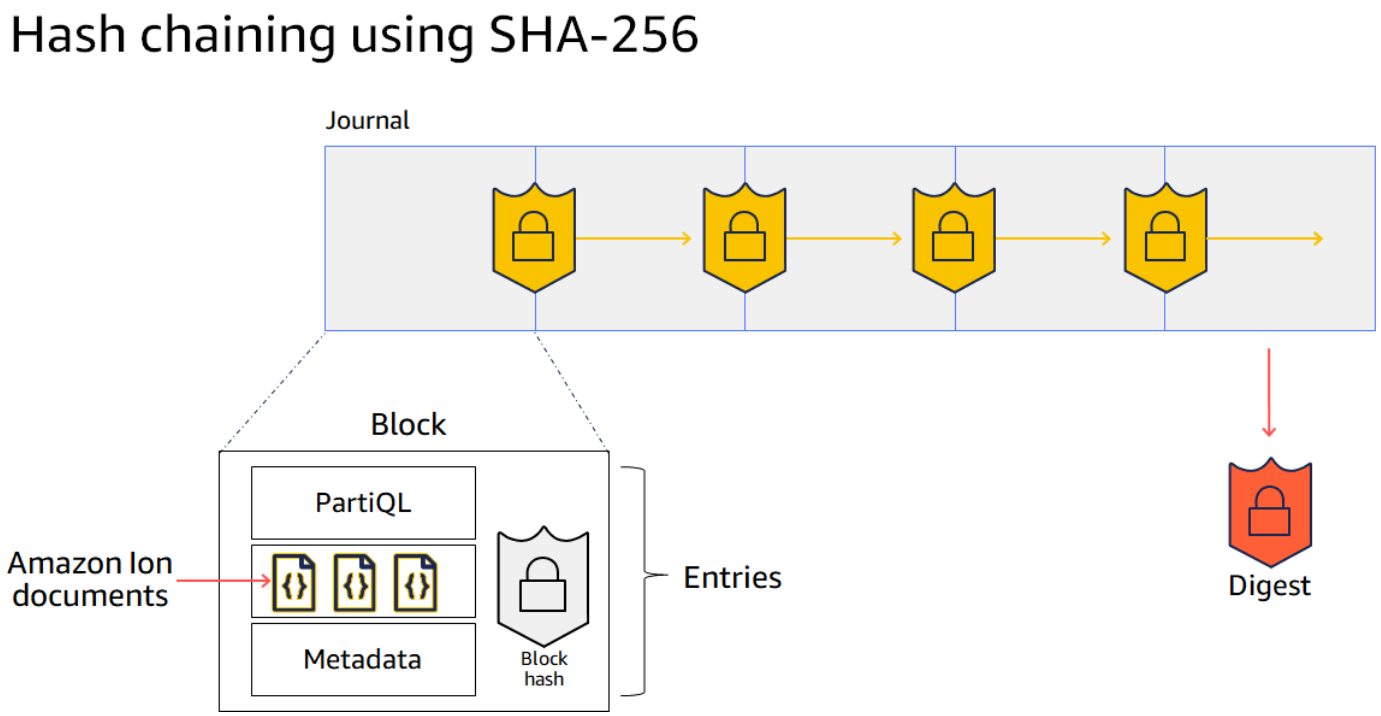


El diagrama muestra que las transacciones se registran en el diario como bloques que se encadenan en hash para su verificación. Cada bloque tiene un número de diario que especifica su dirección.

Verificable criptográficamente

Los bloques del diario se secuencian y encadenan mediante técnicas de hash criptográfico similares al blockchain. QLDB emplea la cadena hash del diario para proporcionar integridad a los datos transaccionales mediante un método de verificación criptográfica. Con un resumen (un valor de hash que representa la cadena de hash completa de una diario en un momento dado) y una prueba de auditoría de Merkle (un mecanismo que demuestra la validez de cualquier nodo de un árbol de hash binario), es posible comprobar que no se han producido cambios involuntarios en sus datos en ningún momento.

El siguiente diagrama muestra un resumen que abarca toda la cadena de hash de un diario en un momento dado.



En este diagrama, los bloques del diario se procesan mediante la función hash criptográfica SHA-256 y se encadenan secuencialmente a los bloques siguientes. Cada bloque contiene entradas que incluyen sus documentos de datos, metadatos y las instrucciones PartiQL ejecutadas en la transacción.

Para obtener más información, consulte Verificación de datos en Amazon QLDB.

Similar a SQL y flexible en cuanto a documentos

QLDB emplea PartiQL como lenguaje de consulta y Amazon Ion como modelo de datos orientado a documentos. PartiQL es un lenguaje de consulta de código abierto compatible con SQL que se ha ampliado para funcionar con Ion. PartiQL le permite insertar, consultar y administrar sus datos con operadores SQL conocidos. Al consultar documentos planos, la sintaxis es la misma que cuando se usa SQL para consultar tablas relacionales. Para obtener más información acerca de la implementación de PartiQL en QLDB, consulte Referencia de PartiQL de Amazon QLDB.

Amazon Ion es un superconjunto de JSON. Ion es un formato de datos de código abierto basado en documentos que le brinda la flexibilidad de almacenar y procesar datos estructurados, semiestructurados y anidados. Para obtener más información sobre Ion en QLDB, consulte Referencia del formato de datos de Amazon Ion en Amazon QLDB.

Para ver una comparación de alto nivel de los componentes y características principales de las bases de datos relacionales tradicionales con QLDB, consulte De relacional a libro mayor.

Herramientas de código abierto para desarrolladores

Para simplificar el desarrollo de aplicaciones, QLDB proporciona controladores de código abierto en distintos lenguajes de programación. Puede usar estos controladores para interactuar con la API de datos transaccionales ejecutando instrucciones PartiQL en un libro mayor y procesando los resultados de dichas instrucciones. Para obtener más información y tutoriales sobre los lenguajes de controlador compatibles actualmente, consulte Introducción al controlador Amazon QLDB.

Amazon Ion también proporciona bibliotecas de cliente que procesan los datos de Ion por usted. Para ver guías para desarrolladores y ejemplos de código sobre el procesamiento de datos de Ion, consulte la documentación de Amazon Ion en GitHub.

Sin servidor y con alta disponibilidad

QLDB está totalmente gestionado y es una solución sin servidor y de alta disponibilidad. El servicio escala automáticamente para satisfacer la demanda de su aplicación. No es necesario aprovisionar instancias ni capacidad. QLDB replica varias copias de sus datos dentro de una zona de disponibilidad y entre zonas de disponibilidad de una Región de AWS.

Nivel empresarial

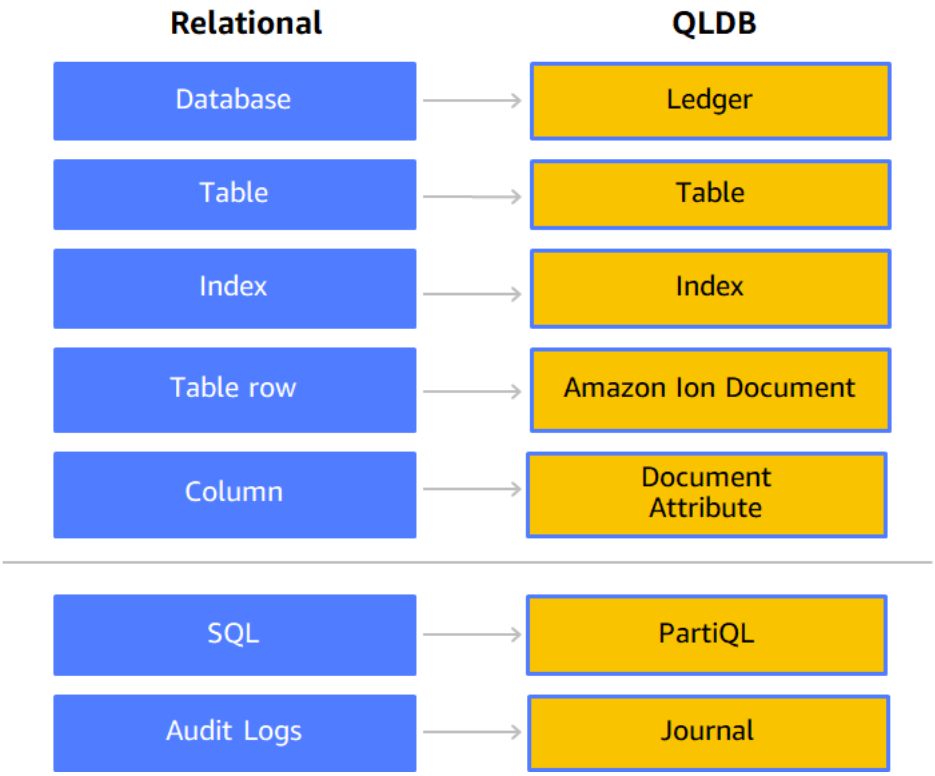
Las transacciones de QLDB son totalmente compatibles con las propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID). QLDB emplea control de concurrencia optimista (OCC) y las transacciones operan con serialización total, el nivel más alto de aislamiento. Esto significa que no hay riesgo de que se produzcan lecturas fantasma, lecturas sucias, errores de escritura u otros problemas de concurrencia similares. Para obtener más información, consulte Modelo de concurrencia de Amazon QLDB.

De relacional a libro mayor

Si es desarrollador de aplicaciones, es posible que tenga alguna experiencia con el sistema de administración de bases de datos relacionales (RDBMS, por sus siglas en inglés) y con el lenguaje de consulta estructurada SQL. Cuando comience a usar Amazon QLDB, observará numerosas similitudes. En los aspectos más avanzados, también encontrará nuevas y potentes funciones que QLDB ha creado sobre la base del RDBMS. En esta sección se describen los componentes y las

operaciones bases de datos habituales, comparando y contrastándolos con sus equivalentes en QLDB.

El siguiente diagrama muestra los constructos de mapeo de los componentes principales entre un RDBMS tradicional y Amazon QLDB.



La siguiente tabla muestra las principales similitudes y diferencias de alto nivel de las características operativas integradas entre un RDBMS tradicional y QLDB.

Operación	RDBMS	QLDB
Creación de tablas	Instrucción `CREATE TABLE` que define todos los nombres de columnas y tipos de datos	Instrucción `CREATE TABLE` que no define ningún atributo de tabla o tipo de datos para permitir contenido abierto y sin esquema
Creación de índices	`CREATE INDEX` instrucción	Instrucción `CREATE INDEX` para cualquier campo de nivel superior de una tabla

Operación	RDBMS	QLDB
Inserción de datos	Instrucción `INSERT` que especifica los valores de una nueva fila o tupla que se ajusta al esquema definido en la tabla	Instrucción `INSERT` que especifica los valores de un documento nuevo en cualquier formato de Amazon Ion válido, independientemente de los documentos existentes en la tabla
Consulta de datos	`SELECT-FROM-WHERE` instrucción	Instrucción `SELECT-FROM-WHERE` con la misma sintaxis que SQL al consultar documentos planos
Actualización de datos	`UPDATE-SET-WHERE` instrucción	Instrucción `UPDATE-SET-WHERE` con la misma sintaxis que SQL al actualizar documentos planos
Eliminación de datos	`DELETE-FROM-WHERE` instrucción	Instrucción `DELETE-FROM-WHERE` con la misma sintaxis que SQL al eliminar documentos planos
Datos anidados y semiestructurados	Solo filas planas o tuplas	Documentos que pueden contener cualquier dato estructurado, semiestructurado o anidado compatibles con el formato de datos Amazon Ion y el lenguaje de consultas PartiQL
Consulta de metadatos	No hay metadatos integrados	Instrucción `SELECT` que consulta desde la vista de confirmación integrada de una tabla

Operación	RDBMS	QLDB
Consultar el historial de revisiones	No hay historial de datos integrado	Instrucción `SELECT` que consulta desde la función de historial integrada
Verificación criptográfica	Sin criptografía ni inmutabilidad integradas	API que devuelven un resumen de un diario y una prueba que verifica la integridad de cualquier revisión de un documento relativa a dicho resumen

Para ver una descripción general de los conceptos básicos y la terminología QLDB, consulte Conceptos clave.

Para obtener información detallada sobre el proceso de creación, consulta y administración de datos en un libro mayor, consulte Uso de datos e historial.

Conceptos básicos y terminología de Amazon QLDB

En esta sección se proporciona información general sobre los principales conceptos y la terminología de Amazon QLDB, incluida la estructura del libro mayor y la forma en que dicho libro mayor gestiona los datos. Como base de datos de libro mayor, QLDB se diferencia de otras bases de datos orientadas a documentos en los siguientes conceptos clave.

Temas

Modelo de objeto de datos QLDB

El modelo de objeto de datos fundamental de Amazon QLDB se describe de la siguiente manera:

1. Libro mayor

   El primer paso es crear un libro mayor, que es el tipo de recurso de AWS principal en QLDB. Para aprender a crear un libro mayor, consulte Paso 1: crear un nuevo libro mayor en Introducción a la consola o Operaciones básicas de libros mayores de Amazon QLDB.

   Para los modos de permisos `ALLOW_ALL` y `STANDARD` del libro mayor, debe crear políticas de AWS Identity and Access Management (IAM) que concedan permisos para ejecutar operaciones de API en este recurso de libro mayor.

   Formato de ARN de libro mayor:

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

2. Secuencia y tablas

   Para empezar a escribir datos en un libro mayor de QLDB, antes debe crear una tabla con una instrucción CREATE TABLE básica. Los datos del libro mayor son revisiones de documentos consignados en el diario del libro mayor. Las revisiones de los documentos se consignan en el libro mayor en el contexto de tablas definidas por el usuario. En QLDB, la tabla representa una vista materializada de una colección de revisiones de documentos del diario.

   En el modo de permisos `STANDARD` del libro mayor, deberá crear políticas de IAM que concedan permisos para ejecutar instrucciones PartiQL en este recurso de tabla. Con los permisos del recurso de tabla podrá ejecutar instrucciones para acceder al estado actual de la tabla. También puede consultar el historial de revisiones de la tabla usando la función integrada `history()`.

   Formato de ARN de tabla:

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

   Para obtener más información sobre cómo conceder permisos en un libro mayor y sus recursos asociados, consulte Cómo funciona Amazon QLDB con IAM.

3. Documentos

Las tablas se componen de revisiones de Documentos de QLDB, que son conjuntos de datos en formato `struct` de Amazon Ion. La revisión de un documento representa una versión única de un diario de documentos identificados mediante una ID de documento único.

QLDB almacena el historial de cambios completo de sus documentos consignados. La tabla le permite consultar el estado actual de sus documentos, mientras que la función `history()` le permite consultar todo el historial de revisiones de los documentos de una tabla. Para obtener más información sobre la consulta y la escritura de revisiones, consulte Uso de datos e historial.

4. Catálogo de sistema

   Cada libro mayor también proporciona un recurso de catálogo definido por el sistema que puede consultar para obtener una lista de todas las tablas e índices de un libro mayor. En el modo de permisos `STANDARD` de un libro mayor, necesita el permiso `qldb:PartiQLSelect` de este recurso de catálogo para hacer lo siguiente:

   - Ejecutar instrucciones `SELECT` en la tabla del catálogo del sistema information_schema.user_tables.
   - Consultar la información de tablas e índices en la página de detalles del libro mayor de la consola de QLDB.
   - Ver la lista de tablas e índices en el editor PartiQL de la consola de QLDB.

   Formato de ARN de catálogo:

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

Transacciones de diario primero

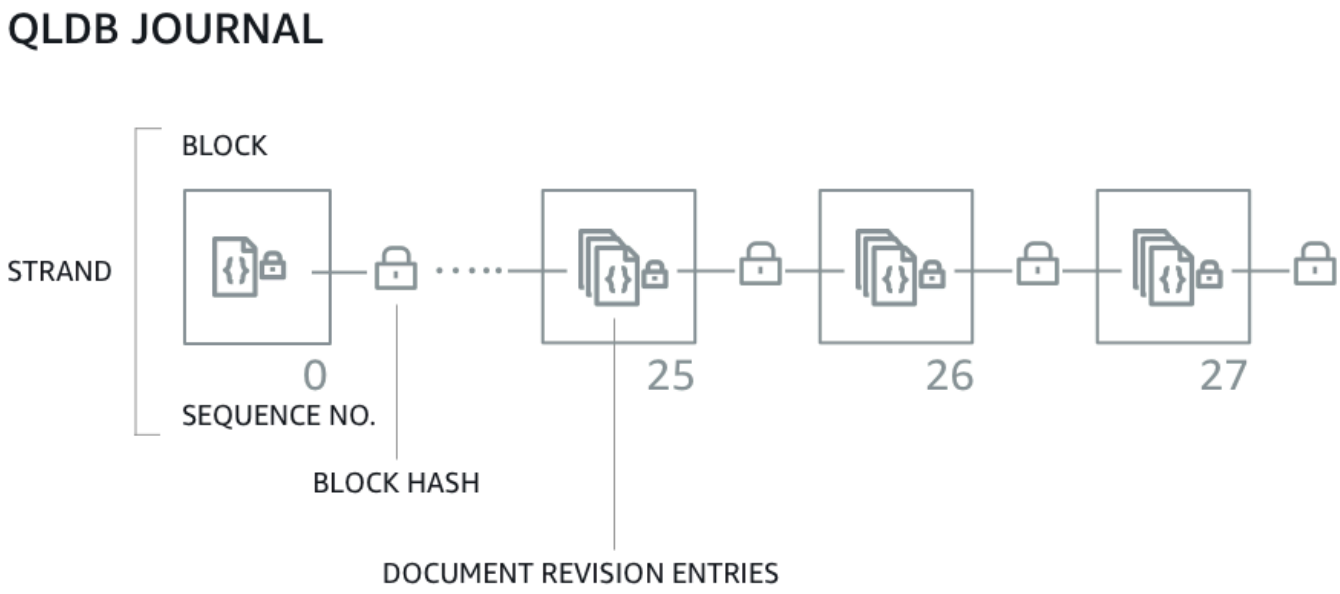
Cuando una aplicación lee o escribe datos en un libro mayor de QLDB, lo hace mediante una transacción de base de datos. Todas las transacciones están sujetas a los límites definidos en Cuotas y límites de Amazon QLDB. En una transacción, QLDB realiza los siguientes pasos:

1. Lee el estado actual de los datos del libro mayor.
2. Lleva a cabo las instrucciones indicadas en la transacción y, a continuación, comprueba si existen conflictos mediante control de concurrencia optimista (OCC) para garantizar un aislamiento totalmente serializable.

3. Si no se encuentra ningún conflicto de OCC, devuelve los resultados de la transacción de la siguiente manera:
   - En el caso de las lecturas, devuelve el conjunto de resultados y consigna las instrucciones `SELECT` en el diario únicamente como anexos.
   - En el caso de las escrituras, confirma en el diario todas las actualizaciones, eliminaciones o datos recién insertados únicamente como anexos.

El libro mayor contiene un historial completo e inmutable de todos los cambios en los datos. QLDB escribe un bloque encadenado en el diario de una transacción. Cada bloque contiene objetos de entrada que representan las revisiones del documento que inserta, actualiza y elimina, junto con las instrucciones PartiQL que confirmaron dichas acciones.

El siguiente diagrama ilustra la estructura de este diario.



El diagrama muestra que las transacciones se registran en el diario como bloques que contienen entradas de revisión de documentos. Cada bloque está codificado y encadenado a los siguientes bloques para su verificación. Cada bloque tiene un número de diario que especifica su dirección dentro de la cadena.

> **Note**
>
> En Amazon QLDB, una cadena es una partición del diario del libro mayor. Actualmente, QLDB admite diarios con una sola cadena.

Para obtener información sobre el contenido de los datos en un bloque, consulte Contenido del diario en Amazon QLDB.

Consulta de sus datos

El objetivo de QLDB es abordar las necesidades de las cargas de trabajo de procesamiento de transacciones online (OLTP) de alto rendimiento. El libro mayor proporciona vistas en tabla consultables de sus datos en función de la información de transacciones consignada en el diario. Una vista de tabla en QLDB es un subconjunto de los datos de una tabla. Las vistas se mantienen en tiempo real, por lo que siempre están disponibles para que las aplicaciones puedan consultarlas.

Puede consultar las siguientes vistas definidas por el sistema usando instrucciones `SELECT` de PartiQL:

- Usuario: la última revisión activa solo de los datos escritos en la tabla (es decir, el estado actual de los datos de usuario). Esta es la vista predeterminada en QLDB.
- Confirmada: la última revisión activa de sus datos de usuario y de los metadatos generados por el sistema. Esta es la tabla completa definida por el sistema que corresponde directamente a su tabla de usuario.

Además de estas vistas consultables, puede consultar el historial de revisiones de sus datos usando la función integrada Función de historial. La función de historial devuelve los datos de usuario y los metadatos asociados en el mismo esquema que la vista confirmada.

Almacenamiento de datos

Existen dos tipos de almacenamiento de datos en QLDB:

- Almacenamiento de secuencias: el espacio en disco que ocupa el diario de un libro mayor. El diario es un anexo que contiene el historial completo, inmutable y verificable de todos los cambios realizados en los datos.

- Almacenamiento indexado: el espacio en disco que ocupan las tablas, índices e historial indexado de un libro mayor. El almacenamiento indexado consiste en datos de libro mayor optimizados para consultas de alto rendimiento.

Una vez confirmados los datos en el diario, se materializan en las tablas que ha definido. Estas tablas están optimizadas para poder realizar consultas más rápidas y eficaces. Cuando una aplicación usa la API de datos transaccionales para leer los datos, accede a las tablas e índices almacenados en el almacenamiento indexado.

Modelo de API de QLDB

QLDB proporciona dos tipos de API con las que el código de su aplicación puede interactuar:

- Amazon QLDB: la API de administración de recursos de QLDB (también conocida como plano de control). Esta API se usa únicamente para administrar los recursos del libro mayor y para las operaciones de datos no transaccionales. Puede usar estas operaciones para crear, eliminar, describir, enumerar y actualizar libros mayores. También puede verificar los datos criptográficamente y exportar o transmitir bloques de diarios.
- Sesión de Amazon QLDB: la API de datos transaccionales de QLDB. Puede usar esta API para ejecutar transacciones de datos en un libro mayor con instrucciones PartiQL.

> **Important**
>
> En lugar de interactuar directamente con esta API de sesión de QLDB, recomendamos usar el controlador QLDB o el intérprete de comandos QLDB para ejecutar transacciones de datos en un libro mayor.
>
> - Si trabaja con un SDK de AWS, use el controlador QLDB. El controlador proporciona una capa de abstracción de alto nivel sobre la API de datos de sesión de QLDB, y gestiona la operación `SendCommand` por usted. Para obtener más información y ver una lista de los lenguajes de programación compatibles, consulte Introducción al controlador.
> - Si está trabajando con la AWS CLI, use el intérprete de comandos QLDB. El intérprete de comandos es una interfaz de línea de comandos que usa el controlador QLDB para interactuar con un libro mayor. Para obtener información, consulte Uso del intérprete de comandos de Amazon QLDB (solo API de datos).

Para obtener más información sobre estas operaciones de la API, consulte Referencia de la API de Amazon QLDB.

Pasos siguientes

Para aprender a usar un libro mayor con sus datos, consulte Trabajar con datos e historial en Amazon QLDB y siga los ejemplos que describen el proceso de creación de tablas, inserción de datos y ejecución de consultas básicas. Esta guía explica en detalle el funcionamiento de estos conceptos con ejemplos de consultas y datos de muestra para ofrecer un mayor contexto.

Para empezar rápidamente con un ejemplo de tutorial de aplicación que usa la consola QLDB, consulte Introducción a la consola de Amazon QLDB.

Para consultar una lista de los términos y definiciones clave descritos en esta sección, acceda a Glosario de Amazon QLDB.

Contenido del diario en Amazon QLDB

En Amazon QLDB, un diario es un registro transaccional inmutable que almacena el historial completo y verificable de todos los cambios en los datos. El diario es solamente un anexo, y está compuesto por un conjunto de bloques secuenciados y encadenados que contienen los datos confirmados y otros metadatos del sistema. QLDB escribe un bloque encadenado en el diario de una transacción.

En esta sección se proporciona un ejemplo de un bloque de diarios con datos de muestra y se describe el contenido del bloque.

Temas

Ejemplo de bloque

Un bloque de diarios contiene metadatos de transacciones junto con entradas que representan las revisiones del documento confirmadas en la transacción, así como las instrucciones PartiQL que las han confirmado.

El siguiente es un ejemplo de un bloque con datos de muestra.

> **Note**
>
> Este ejemplo de bloque se ofrece solo con fines informativos. Los hash que se muestran no son valores hash calculados reales.

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

En el campo `revisions`, es posible que algunos objetos de revisión contengan solo un valor `hash` y ningún otro atributo. Son revisiones del sistema exclusivamente internas que no contienen datos de usuario. Los hash de estas revisiones forman parte de la cadena de hash completa del diario, necesaria para la verificación criptográfica.

Contenidos del bloque

Un bloque de diarios cuenta con los siguientes campos:

`blockAddress`

La ubicación del bloque en el diario. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

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

`transactionId`

El ID único de la transacción que confirmó el bloque.

`blockTimestamp`

La fecha y hora de confirmación del bloque en el diario.

`blockHash`

El valor de hash de 256 bits que representa ese bloque único. Es el hash de la concatenación de `entriesHash` y `previousBlockHash`.

`entriesHash`

El hash que representa todas las entradas del bloque, incluidas las entradas del sistema únicamente internas. Se trata del hash raíz del árbol de Merkle, en el que los nodos de las hojas están compuestos por todos los hash de `entriesHashList`.

`previousBlockHash`

El hash del anterior bloque encadenado del diario.

`entriesHashList`

La lista de hash que representan cada entrada del bloque. Esta lista puede incluir los siguientes hash de entrada:

- El hash de Ion que representa `transactionInfo`. Este valor se calcula tomando el hash de Ion de toda la estructura de `transactionInfo`.
- Se trata del hash raíz del árbol de Merkle, en el que los nodos de las hojas están compuestos por todos los hash de `revisions`.
- El hash de Ion que representa `redactionInfo`. Este hash solo existe en los bloques confirmados mediante una transacción de redacción. Este valor se calcula tomando el hash de Ion de toda la estructura de `redactionInfo`.
- Hashes que representan metadatos del sistema únicamente internos. Es posible que estos hashes no existan en todos los bloques.

`transactionInfo`

Estructura de Amazon Ion que contiene información sobre las instrucciones de la transacción en la que se confirmó el bloque. Esta estructura tiene los campos siguientes:

- `statements`: la lista de instrucciones PartiQL y el `startTime` en que comenzaron a ejecutarse. Cada instrucción tiene un hash de `statementDigest` necesario para calcular el hash de la estructura de `transactionInfo`.
- `documents`: los ID de los documentos que se actualizaron con las instrucciones. Cada documento incluye el `tableName` e `tableId` al que pertenece, así como el índice de cada instrucción que lo actualizó.

`revisions`

La lista de revisiones de documentos confirmadas en el bloque. Cada estructura de revisión contiene todos los campos de la vista confirmada de la revisión.

También puede incluir hashes que representen revisiones del sistema exclusivamente internas y que formen parte de la cadena de hash completa de un diario.

Revisiones redactadas

En Amazon QLDB, una instrucción `DELETE` solo elimina un documento de forma lógica al crear una nueva revisión que lo marca como eliminado. QLDB también permite realizar una operación de edición de datos para eliminar permanentemente las revisiones de documentos inactivos del historial de una tabla.

La operación de redacción elimina solo los datos de usuario de la revisión especificada, sin alterar el diario ni los metadatos del documento. Esto mantiene la integridad general de los datos del libro mayor. Para obtener más información y ver un ejemplo de operación de redacción, consulte Editar revisiones de documentos.

Ejemplo de revisión redactada

Considere el ejemplo de bloque anterior. En este bloque, supongamos que redacta la revisión con ID de documento `HwVFkn8IMRa0xjze5xcgga` y número de versión `0`.

Una vez finalizada la redacción, los datos de usuario de la revisión (representados por la estructura `data`) se sustituyen por un nuevo campo `dataHash`. El valor de este campo es el hash de Ion de la estructura `data` eliminada. Como resultado, el libro mayor mantiene la integridad general de sus datos y sigue siendo verificable criptográficamente mediante las operaciones de la API de verificación existentes.

El siguiente ejemplo de revisión muestra los resultados de esta redacción, con el nuevo campo `dataHash` resaltado en *`cursiva roja`*.

> **Note**
>
> Este ejemplo de revisión se ofrece solo con fines informativos. Los hash que se muestran no son valores hash calculados reales.

```
...
{
  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 también añade un nuevo bloque al diario con la solicitud finalizada de redacción. En este bloque se incluye una entrada `redactionInfo` adicional que contiene una lista de las revisiones redactadas en la transacción, tal como se muestra en el ejemplo siguiente.

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

```
...
```

Aplicación de muestra

Para ver un ejemplo de código Java que valida la cadena de hash de un diario usando datos exportados, consulte el repositorio de GitHub aws-samples/amazon-qldb-dmv-sample-java. Esta aplicación de ejemplo incluye los siguientes archivos de clase:

- ValidateQldbHashChain.java: contiene un código tutorial que exporta bloques de diarios de un libro mayor y emplea los datos exportados para validar la cadena de hash entre bloques.
- JournalBlock.java: contiene un método denominado `verifyBlockHash()` que muestra cómo calcular cada componente de hash individual de un bloque. Este método se invoca mediante el código del tutorial de `ValidateQldbHashChain.java`.

Para obtener instrucciones sobre cómo descargar e instalar esta aplicación de ejemplo completa, consulte Instalación de la aplicación de ejemplo Java de Amazon QLDB. Antes de ejecutar el código del tutorial, asegúrese de seguir los pasos 1 a 3 de Tutorial de Java para configurar un libro mayor y cargarlo con datos de ejemplo.

Véase también

Para obtener más información sobre diarios en QLDB, consulte los siguientes temas:

- Exportación de datos de diarios desde Amazon QLDB: para saber cómo exportar datos del diario a Amazon Simple Storage Service (Amazon S3).
- Transmisión de datos de diarios desde Amazon QLDB: para saber cómo transmitir datos del diario a Amazon Kinesis Data Streams.
- Verificación de datos en Amazon QLDB: para obtener más información sobre la verificación criptográfica de los datos del diario.

Glosario de Amazon QLDB

A continuación, se muestran las definiciones de los términos clave que es posible que encuentre en Amazon QLDB.

bloque | resumen | documento | ID del documento | revisión de documento | entrada | campo | índice | almacenamiento indexado | diario | bloque del diario | almacenamiento del diario | cadena del diario | tip del diario | libro mayor | prueba | revisión | sesión | cadena | tabla | vista de tabla | ver

bloque

Un objeto que se asigna al diario durante una transacción. Una sola transacción escribe un bloque en el diario, por lo que un bloque solo se puede asociar a una transacción. Un bloque contiene entradas que representan las revisiones del documento que se registraron en la transacción y las instrucciones PartiQL que las confirmaron.

Cada bloque también tiene un valor hash para su verificación. Un hash de bloque se calcula a partir de los hashes de entrada de ese bloque combinados con el hash del bloque encadenado anterior.

resumen

Un valor de hash de 256 bits que representa de forma única todo el historial de revisiones de documentos del libro mayor en un momento dado. El hash de resumen se calcula a partir de la cadena de hash completa del diario a partir del último bloque registrado en el diario en ese momento.

QLDB le permite generar un resumen como un archivo de salida seguro. Luego, puede usar ese archivo de salida para verificar la integridad de las revisiones de sus documentos en relación con ese hash.

documento

Conjunto de datos en formato `struct` de Amazon Ion que se puede insertar, actualizar y eliminar en una tabla. Un documento QLDB puede tener datos estructurados, semiestructurados, anidados y sin esquema.

ID del documento

Número único universal (UUID) que QLDB asigna a cada documento que inserta en una tabla. Este identificador es un número de 128 bits que se representa en una cadena alfanumérica codificada en Base62 con una longitud fija de 22 caracteres.

revisión de documento

La revisión de un documento representa una versión única de una secuencia de documentos identificados mediante un ID de documento único. Una revisión incluye tanto los datos de usuario (es decir, los datos que escribió en la tabla) como los metadatos generados por el sistema. Cada

revisión se asocia con una tabla y se identifica de forma única mediante una combinación del identificador del documento y un número de versión de base cero.

entrada

Objeto dentro de un bloque. Las entradas representan las revisiones de documentos que se insertan, actualizan y eliminan en una transacción, junto con las instrucciones de PartiQL que las confirmaron.

Cada entrada también tiene un valor hash para su verificación. El hash de entrada se calcula a partir de los hashes de revisión o de los hashes de instrucciones incluidos en esa entrada.

campo

Un par nombre-valor que compone cada atributo de un documento de QLDB. El nombre es un token de símbolo y el valor no tiene restricciones.

índice

Estructura de datos que se puede crear en una tabla para optimizar el rendimiento de las operaciones de recuperación de datos. Para obtener información sobre los índices de QLDB, consulte CREATE INDEX en la referencia de PartiQL de Amazon QLDB.

almacenamiento indexado

El espacio en disco que ocupan las tablas, índices e historial indexado de un libro mayor. El almacenamiento indexado consiste en datos de libro mayor optimizados para consultas de alto rendimiento.

diario

El conjunto hash encadenado de todos los bloques confirmados en su libro mayor. El diario es un archivo de solo anexado que representa un historial completo e inmutable de todos los cambios en su libro mayor.

bloque del diario

Consulte bloque.

almacenamiento del diario

el espacio en disco que utiliza el diario de un libro mayor.

cadena del diario

Consulte cadena.

tip del diario

El último bloque confirmado en un diario en un momento determinado.

libro mayor

Instancia de un recurso de base de datos de libro mayor de Amazon QLDB. Es el tipo de recurso AWS principal de QLDB. Un libro mayor consiste tanto en el almacenamiento del diario como en el almacenamiento indexado. Una vez guardados los datos del libro mayor en el diario, están disponibles para consultarlos en las tablas de revisiones de documentos de Amazon Ion.

prueba

Lista ordenada de valores hash de 256 bits que QLDB devuelve para un resumen y una revisión de documento determinados. Consiste de los hashes que requiere un modelo de árbol de Merkle para encadenar el hash de la revisión determinada al hash del resumen. La prueba se utiliza para comprobar la integridad de las revisiones relacionadas con el resumen. Para obtener más información, consulte Verificación de datos en Amazon QLDB.

revisión

Consulte revisión de documento.

sesión

Objeto que administra la información sobre las solicitudes de transacciones de datos y las respuestas desde y hacia el libro mayor. Una sesión activa (aquella en la que se ejecuta activamente una transacción) representa una conexión única a un libro mayor. QLDB admite una transacción en ejecución activa por sesión.

cadena

Una partición de un diario. Actualmente, QLDB admite diarios con una sola cadena.

tabla

Vista materializada de una colección desordenada de revisiones de documentos que están confirmadas en el diario del libro mayor.

vista de tabla

Un subconjunto consultable de los datos de una tabla, basado en las transacciones confirmadas en el diario. En una instrucción de PartiQL, una vista se designa con un calificador de prefijo (que comienza con `_ql_`) para el nombre de una tabla.

Puede consultar las siguientes vistas definidas por el sistema usando instrucciones `SELECT` de PartiQL:

- Usuario: la última revisión activa solo de los datos escritos en la tabla (es decir, el estado actual de los datos de usuario). Esta es la vista predeterminada en QLDB.
- Confirmada: la última revisión activa de sus datos de usuario y de los metadatos generados por el sistema. Esta es la tabla completa definida por el sistema que corresponde directamente a su tabla de usuario. Por ejemplo: `_ql_committed_`*`TableName`*.

ver

Consulte vista de tabla.

Acceso a Amazon QLDB

Puede acceder a Amazon QLDB mediante AWS Management Console la API, la AWS CLI() o AWS Command Line Interface la QLDB. En las secciones siguientes, se describe cómo utilizar estas opciones y los requisitos previos para usarlas.

Requisitos previos

Antes de poder acceder a la QLDB, debe configurar Cuenta de AWS una si aún no lo ha hecho.

Temas

Inscríbase en una Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en https://aws.amazon.com/ y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión AWS Management Consolecomo propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Signing in as the root user en la Guía del usuario de AWS Sign-In .

2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .

2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la Guía del AWS IAM Identity Center usuario.

Iniciar sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, utilice la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte Iniciar sesión en el portal de AWS acceso en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .
2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center .

Gestione los permisos de QLDB en IAM

Para obtener información sobre el uso de AWS Identity and Access Management (IAM) para administrar los permisos de QLDB para los usuarios, consulte. Cómo funciona Amazon QLDB con IAM

Conceder acceso programático (opcional)

Los usuarios necesitan acceso programático si quieren interactuar con AWS personas ajenas a. AWS Management Console La forma de conceder el acceso programático depende del tipo de usuario que acceda. AWS

Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Usa credenciales temporales para firmar las solicitudes programáticas a los AWS CLI AWS SDK o las API. AWS	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Configuración del uso AWS IAM Identity Center en AWS CLI la Guía del AWS Command Line Interface usuario.

¿Qué usuario necesita acceso programático?	Para	Mediante
		• Para obtener AWS información sobre los SDK, las herramientas y AWS las API, consulte la autenticación del IAM Identity Center en la Guía de referencia de AWS los SDK y las herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas a los AWS SDK o las AWS CLI API. AWS	Siga las instrucciones de Uso de credenciales temporales con AWS recursos de la Guía del usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales de larga duración para firmar las solicitudes programáticas a los AWS CLI AWS SDK o las API. AWS	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Autenticación con credencia les de usuario de IAM en la Guía del usuario.AWS Command Line Interface • Para obtener informaci ón AWS sobre los SDK y las herramientas, consulte Autenticarse con credencia les de larga duración en la Guía de referencia de los AWS SDK y las herramien tas. • Para obtener informaci ón AWS sobre las API, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

Cómo acceder a Amazon QLDB

Tras completar los requisitos previos para configurar una Cuenta de AWS, consulte los siguientes temas para obtener más información sobre cómo acceder a la QLDB:

- Mediante la consola
- Uso de la AWS CLI (solo la API de administración)
- Uso del intérprete de comandos de Amazon QLDB (solo API de datos)
- Uso de la API

Acceso a Amazon QLDB mediante la consola

Puede acceder a la AWS Management Console QLDB de Amazon en https://console.aws.amazon.com/qldb.

Puede utilizar la consola para hacer lo siguiente en QLDB:

- Cree, elimine, describa y enumere los libros mayores.
- Ejecute instrucciones PartiQL mediante el editor de PartiQL.
- Administre las etiquetas de los recursos de QLDB.
- Verifique criptográficamente los datos del diario.
- Exporte o transmita bloques de diario.

Para obtener información sobre cómo crear un libro mayor de Amazon QLDB y configurarlo con ejemplos de datos de aplicación, consulte Introducción a la consola de Amazon QLDB.

Referencia rápida del editor de PartiQL

Amazon QLDB admite un subconjunto de PartiQL como lenguaje de consulta y Amazon Ion como formato de datos orientado a documentos. Para obtener una guía completa e información más detallada sobre la implementación de PartiQL en QLDB, consulte Referencia de PartiQL de Amazon QLDB.

Los siguientes temas proporcionan una descripción general de referencia rápida sobre cómo utilizar PartiQL en QLDB.

Temas

- Consejos rápidos sobre PartiQL en QLDB
- Comandos
- Vistas definidas por el sistema
- Reglas básicas de sintaxis
- Atajos de teclado del editor PartiQL

Consejos rápidos sobre PartiQL en QLDB

El siguiente es un breve resumen de los consejos y las prácticas recomendadas para trabajar con PartiQL en QLDB:

- Conozca los límites de concurrencia y de transacciones: todas las instrucciones, incluidas las consultas `SELECT`, están sujetas a conflictos de control de concurrencia optimista (OCC) y a límites de transacciones, incluyendo un tiempo de espera de transacción de 30 segundos.
- Utilice índices: utilice índices de cardinalidad alta y ejecute consultas dirigidas para optimizar sus instrucciones y evitar tener que escanear tablas completas. Para obtener más información, consulte Optimizar el rendimiento de las consultas.
- Utilice predicados de igualdad: las búsquedas indexadas requieren un operador de igualdad (`=` o `IN`). Los operadores de desigualdad (`<`, `>`, `LIKE`, `BETWEEN`) no cumplen los requisitos para las búsquedas indexadas y dan como resultado escaneos de tablas completas.
- Utilice únicamente combinaciones internas: QLDB solo admite combinaciones internas. Como práctica recomendada, combine los campos que estén indexados para cada tabla que vaya a unir. Elija índices de cardinalidad alta tanto para los criterios de unión como para los predicados de igualdad.

Comandos

La QLDB admite los siguientes comandos de PartiQL.

Lenguaje de definición de datos (DDL)

Comando	Descripción
CREATE INDEX	Crea un índice para un campo de documento de nivel superior de una tabla.
CREATE TABLE	Crea una tabla.
DROP INDEX	Elimina un índice de una tabla.
DROP TABLE	Desactiva una tabla existente.
UNDROP TABLE	Reactiva una tabla inactiva.

Lenguaje de manipulación de datos (DML)

Comando	Descripción
DELETE	Marca un documento activo como eliminado mediante la creación de una nueva revisión final del documento.
FROM (INSERT, REMOVE o SET)	Semánticamente igual que UPDATE.
INSERT	Añade uno o más documentos a una tabla.
SELECT	Recupera datos de una o más tablas.
UPDATE	Actualiza, inserta o elimina elementos específicos de un documento.

Ejemplos de instrucción DML

INSERT

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

UPDATE-INSERT

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

UPDATE-REMOVE

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

SELECT – subconsulta correlacionada

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

SELECT – combinación interna

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

SELECT – obtiene la ID del documento con la cláusula BY

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

Vistas definidas por el sistema

QLDB admite las siguientes vistas definidas por el sistema de una tabla.

Visualización	Descripción
table_name	La vista de usuario predeterminada de una tabla que incluye únicamente el estado actual de los datos de usuario.
_ql_committed_ *table_name*	La vista confirmada completa definida por el sistema de una tabla que incluye el estado actual de los datos de usuario y los metadatos generados por el sistema, como el identificador de un documento.
history(*table_name*)	La función de historial integrada que devuelve el historial de revisiones completo de una tabla.

Reglas básicas de sintaxis

La QLDB admite las siguientes reglas básicas de sintaxis para PartiQL.

Carácter	Descripción
'	Las comillas simples indican valores de cadena o nombres de campo en las estructuras de Amazon Ion.
"	Las comillas dobles indican identificadores entre comillas, como una palabra reservada que se usa como nombre de tabla.
`	Los acentos graves indican valores literales de Ion.
.	La notación de puntos permite acceder a los nombres de campo de una estructura principal.
[ ]	Los corchetes definen una `list` de Ion o indican un número ordinal basado en cero para una lista existente.
{ }	Los corchetes definen un `struct` de Ion.
<< >>	Los paréntesis angulares dobles definen una bag de PartiQL, que es una colección desordenada. Se utiliza una bag para insertar varios documentos en una tabla.
Sensibilidad de mayúsculas y minúsculas	Todos los nombres de objetos del sistema de QLDB, incluidos los nombres de campos y tablas, distinguen mayúsculas de minúsculas.

Atajos de teclado del editor PartiQL

El editor PartiQL de la consola de QLDB admite los siguientes atajos de teclado.

Acción	macOS	Windows
Ejecute	Cmd+Return	Ctrl+Enter
Comentario	Cmd+/	Ctrl+/

Acción	macOS	Windows
Clear	Cmd+Shift+Delete	Ctrl+Shift+Delete

Acceso a Amazon QLDB mediante la (solo API AWS CLI de administración)

Puede usar el AWS Command Line Interface (AWS CLI) para controlar varios Servicios de AWS desde la línea de comandos y automatizarlos mediante scripts. Puede usar el AWS CLI para operaciones únicas según sea necesario. También puede usarla para incluir operaciones de Amazon QLDB en scripts de utilidades.

Para acceder a la CLI, necesita un ID de clave de acceso y una clave de acceso secreta. Cuando sea posible, utilice credenciales temporales en lugar de claves de acceso. Las credenciales temporales incluyen un ID de clave de acceso y una clave de acceso secreta, pero, además, incluyen un token de seguridad que indica cuándo caducan las credenciales. Para obtener más información, consulte Uso de credenciales temporales con AWS recursos en la Guía del usuario de IAM.

Para obtener un listado completo y ejemplos de uso de todos los comandos disponibles para QLDB en la AWS CLI, consulte Referencia de comandos de AWS CLI.

> **Note**
>
> AWS CLI Solo admite las operaciones `qldb` de la API de administración que se enumeran en. Referencia de la API de Amazon QLDB Esta API se usa únicamente para administrar los recursos del libro mayor y para las operaciones de datos no transaccionales.
> Para ejecutar transacciones de datos con la API `qldb-session` mediante una interfaz de la línea de comandos, consulte Acceso a Amazon QLDB mediante el intérprete de comandos de QLDB (solo API de datos).

Temas

- Instalación y configuración de la AWS CLI
- Uso del AWS CLI con QLDB

Instalación y configuración de la AWS CLI

Se AWS CLI ejecuta en Linux, macOS o Windows. Para instalarla y configurarla, vea las siguientes instrucciones en la Guía del usuario de AWS Command Line Interface :

1. Instalar o actualizar la versión más reciente del AWS CLI
2. Configuración Rápida

Uso del AWS CLI con QLDB

El formato de la línea de comandos se compone de un nombre de operación de Amazon QLDB seguido de los parámetros de dicha operación. AWS CLI Admite una sintaxis abreviada para los valores de los parámetros, además de JSON.

Utilice la `help` para mostrar todos los comandos disponibles en QLDB:

```
aws qldb help
```

También puede utilizar la `help` para describir un comando específico y obtener más información sobre su uso:

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

Por ejemplo, para crear un libro mayor:

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

Acceso a Amazon QLDB mediante el intérprete de comandos de QLDB (solo API de datos)

Amazon QLDB proporciona un intérprete de comandos de línea de comandos para la interacción con la API de datos transaccionales. Con el intérprete de comandos QLDB, puede ejecutar instrucciones PartiQL en datos del libro mayor.

La última versión de este intérprete de comandos está escrita en Rust y es de código abierto en el repositorio awslabs/amazon-qldb-shell de GitHub en la rama `main` predeterminada. La versión de Python (v1) también está disponible para su uso en el mismo repositorio de la rama `master`.

> **Note**
>
> El intérprete de comandos de Amazon QLDB solo admite la API de datos transaccionales `qldb-session`. Esta API se usa solo para ejecutar instrucciones PartiQL en un libro mayor de QLDB.
>
> Para interactuar con las operaciones de la API `qldb` de administración mediante una interfaz de línea de comandos, consulte Acceso a Amazon QLDB mediante la (solo API AWS CLI de administración).

Esta herramienta no está destinada a incorporarse a una aplicación ni a adoptarse con fines de producción. El objetivo de esta herramienta es permitirle experimentar rápidamente con QLDB y PartiQL.

En las secciones siguientes se describe cómo comenzar a utilizar la sincronización del intérprete de comandos de QLDB.

Temas

Requisitos previos

Antes de empezar a usar el intérprete de comandos de QLDB, debe hacer lo siguiente:

1. Siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Estas incluyen las siguientes:

    1. Regístrese en AWS.

2. Cree un usuario con los permisos de QLDB adecuados.
3. Conceda acceso programático de desarrollo.

2. Configure sus credenciales AWS y su Región de AWS predeterminada. Para obtener instrucciones, consulte Configuración rápida en la Guía del usuario de AWS Command Line Interface.

   Para ver una lista completa de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.
3. Para cualquier libro mayor en el modo de permisos `STANDARD`, deberá crear políticas de IAM que concedan permisos para ejecutar instrucciones PartiQL en las tablas adecuadas. Para obtener información sobre cómo crear estas políticas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Instalación del intérprete de comandos

Para instalar la versión más reciente del intérprete de comandos de QLDB, consulte el archivo README.md en GitHub. QLDB proporciona archivos binarios precompilados para Linux, macOS y Windows en la sección Lanzamientos del repositorio de GitHub.

Para macOS, el intérprete de comandos se integra con el tap de Homebrew `aws/tap`. Para instalar el intérprete de comandos en macOS con Homebrew, ejecute los siguientes comandos.

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

Configuración

Tras la instalación, el intérprete de comandos carga el archivo de configuración predeterminado que se encuentra en `$XDG_CONFIG_HOME/qldbshell/config.ion` durante la inicialización. En Linux y macOS, este archivo suele aparecer como `~/.config/qldbshell/config.ion`. Si dicho archivo no existe, el intérprete de comandos se ejecuta con la configuración predeterminada.

Puede crear un archivo `config.ion` manualmente después de la instalación. Este archivo de configuración utiliza el formato de datos de Amazon Ion. A continuación se presenta un ejemplo de un archivo `config.ion`.

```
{
```

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

Si `default_ledger` no está establecido en el archivo de configuración, el parámetro `--ledger` es obligatorio para invocar el intérprete de comandos. Para ver una lista completa de opciones de configuración, consulte el archivo README.md en GitHub.

Invocar el intérprete de comandos

Para invocar el intérprete de comandos QLDB en su terminal de línea de comandos para un libro mayor específico, ejecute el siguiente comando. Sustituya *`my-example-ledger`* por el nombre de su libro mayor.

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

Este comando se conecta a su Región de AWS predeterminado. Para especificar la región de forma explícita, puede ejecutar el comando con el parámetro `--region` o `--qldb-session-endpoint`, tal y como se describe en la siguiente sección.

Tras invocar una sesión de intérprete de comandos `qldb`, puede introducir los siguientes tipos de entrada:

- Comandos de intérprete de comandos
- Instrucciones PartiQL únicas en transacciones separadas
- Instrucciones PartiQL múltiples dentro de una transacción

Parámetros de intérprete de comandos

Para obtener una lista completa de los marcadores y opciones disponibles para invocar un intérprete de comandos, ejecute el comando `qldb` con el indicador `--help`, de la siguiente manera.

```
$ qldb --help
```

A continuación se enumeran opciones y marcadores clave del comando `qldb`. Puede añadir estos parámetros opcionales para anular Región de AWS, el perfil de credenciales, el punto de conexión, el formato de los resultados y otras opciones de configuración.

Uso

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

MARCADORES

-h, **--help**

Imprime información de ayuda.

-v, **--verbose**

Configura el nivel de detalle del registro. De forma predeterminada, el intérprete de comandos solo registra los errores. Para aumentar el nivel de detalle, repita este argumento (por ejemplo, `-vv`). El nivel más alto es `-vvv`, que corresponde al nivel de detalle `trace`.

-V, **--version**

Imprime la información de la versión.

OPTIONS

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

Nombre del libro mayor al que se va a conectar. Se trata de un parámetro de intérprete de comandos obligatorio si `default_ledger` no está establecido en el archivo `config.ion`. En este archivo, puede configurar opciones adicionales, como la región.

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

El archivo en el que puede definir cualquier opción de configuración de intérprete de comandos. Para ver los detalles de formato y una lista completa de opciones de configuración, consulte el archivo README.md en GitHub.

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

El formato de salida de los resultados de su consulta. El valor predeterminado es `ion`.

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

La ubicación del perfil de credenciales AWS que se utilizará para la autenticación.

Si no se proporciona, el intérprete de comandos utilizará su perfil predeterminado AWS, que se encuentra en `~/.aws/credentials`.

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

El código Región de AWS del libro mayor de QLDB al que conectarse. Por ejemplo: `us-east-1`.

Si no se proporciona, el intérprete de comandos se conecta a su Región de AWS predeterminada, tal y como se especifica en su perfil AWS.

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

El punto de conexión de la API `qldb-session` usado para conectarse.

Para ver una lista completa de las regiones y puntos de conexión QLDB disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

Referencia de los comandos

Tras invocar una sesión `qldb`, el intérprete de comandos admite las siguientes claves y comandos de base de datos:

Claves de intérprete de comandos

Clave	Descripción de función
Enter	Ejecute la instrucción.
Escape+Enter (macOS, Linux) Shift+Enter (Windows)	Inicia una nueva línea para introducir una instrucción que abarque varias líneas. También puede copiar el texto introducido con varias líneas y pegarlo en el intérprete de comandos. Para obtener instrucciones sobre cómo configurar Option en lugar de Escape como una clave meta en macOS, consulte el sitio de OS X Daily.
Ctrl+C	Cancela el comando actual.
Ctrl+D	Señala el final del archivo (EOF) y sale del nivel actual del intérprete de comandos. Si no está en una transacción activa, sale del

Clave	Descripción de función
	intérprete de comandos. En una transacción activa, anula la transacción.

Comandos de la base de datos del intérprete de comandos

Comando	Descripción de función
`help`	Muestra la información de ayuda.
`begin`	Inicia una transacción.
`start transaction`	
`commit`	Confirma la transacción en el diario del libro mayor.
`abort`	Detiene la transacción y rechaza cualquier cambio que haya realizado.
`exit`	Sale del intérprete de comandos.
`quit`	

> **Note**
>
> Los comandos del intérprete de comandos de QLDB no distingue ninguno entre mayúsculas y minúsculas.

Ejecutar instrucciones individuales

A excepción de los comandos de la base de datos y los metacomandos del intérprete de comandos que aparecen en README.md, el intérprete de comandos interpreta cada comando que se introduce como una instrucción PartiQL independiente. De forma predeterminada, el intérprete de comandos habilita el modo `auto-commit`. Este modo es configurable.

En el modo `auto-commit`, el intérprete de comandos ejecuta implícitamente cada instrucción en su propia transacción y la confirma automáticamente si no se encuentra ningún error. Esto significa que no tiene que ejecutar `start transaction` (ni `begin`) y `commit` manualmente cada vez que ejecute una instrucción.

Administración de transacciones

Como alternativa, el intérprete de comandos de QLDB le permite controlar manualmente las transacciones. Puede ejecutar varias instrucciones dentro de una transacción de forma interactiva o no interactiva agrupando comandos e instrucciones por lotes de forma secuencial.

Transacciones activas

Para ejecutar una transacción interactiva, realice los siguientes pasos.

1. Para iniciar una transacción, introduzca el comando `begin`.

   ```
   qldb> begin
   ```

   Tras iniciar una transacción, el intérprete de comandos muestra la siguiente línea de comandos.

   ```
   qldb *>
   ```

2. A continuación, todas las instrucciones que introduzca se ejecutarán en la misma transacción.
   - Por ejemplo, puede ejecutar una sola instrucción de la siguiente manera.

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

     Tras pulsar Enter, el intérprete de comandos muestra los resultados de la instrucción.
   - También puede introducir varias instrucciones o comandos separados por un delimitador de punto y coma (`;`) de la siguiente manera.

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

3. Para finalizar la transacción, inserte uno de los siguientes comandos.
   - Introduzca el comando `commit` para confirmar la transacción en el diario del libro mayor.

     ```
     qldb *> commit
     ```

- Introduzca el comando `abort` para detener la transacción y rechazar cualquier cambio que haya realizado.

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

Límite de tiempo de espera de transacción

Una transacción interactiva cumple con el límite de tiempo de espera de las transacciones de QLDB. Si no confirma una transacción en un plazo de 30 segundos desde su inicio, QLDB vence automáticamente la transacción y rechaza cualquier cambio realizado durante la transacción.

A continuación, en lugar de mostrar los resultados de la instrucción, el intérprete de comandos muestra un mensaje de error de caducidad y vuelve a la línea de comandos normal. Para volver a intentarlo, debe volver a introducir el comando `begin` para iniciar una nueva transacción.

```
transaction failed after 1 attempts, last error: communication failure:
 Transaction 2UMpiJ5hh7WLjVgEiML0oO has expired
```

Transacciones no interactivas

Puede ejecutar una transacción completa con varias instrucciones agrupando los comandos y las declaraciones de forma secuencial de la siguiente manera.

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

Debe separar cada comando y cada instrucción con un delimitador de punto y coma (`;`). Si alguna de las instrucciones de la transacción no es válida, el intérprete de comandos la rechaza automáticamente. El intérprete de comandos no procederá con ningún extracto posterior que haya introducido.

También puede configurar varias transacciones.

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

Al igual que en el ejemplo anterior, si se produce un error en una transacción, el intérprete de comandos no procederá con ninguna transacción o instrucción posterior que haya introducido.

Si no finaliza una transacción, el intérprete de comandos pasa al modo interactivo y le pide el siguiente comando o instrucción.

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

Salir del intérprete de comandos

Para salir de la sesión de intérprete de comandos `qldb` actual, introduzca el comando `exit` o `quit`, o utilice la combinación de teclas Ctrl+D cuando el intérprete de comandos no esté incluido en una transacción.

```
qldb> exit
$
```

```
qldb> quit
$
```

Ejemplo

Para obtener información sobre cómo escribir instrucciones PartiQL en QLDB, consulte Referencia de PartiQL de Amazon QLDB.

Example

En el siguiente ejemplo, se muestra una secuencia común de comandos básicos.

> **Note**
>
> El intérprete de comandos QLDB ejecuta cada instrucción PartiQL de este ejemplo en su propia transacción.
> En este ejemplo se supone que el libro mayor `test-ledger` ya existe y está activo.

```
$ 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
```

Acceso a Amazon QLDB mediante la API

Puede utilizar los caracteres AWS Management Console y AWS Command Line Interface (AWS CLI) para trabajar de forma interactiva con Amazon QLDB. Sin embargo, para aprovechar al máximo la QLDB, puede escribir el código de la aplicación con un controlador de QLDB o AWS un SDK para interactuar con su libro mayor mediante las API.

El controlador permite que su aplicación interactúe con QLDB mediante la API de datos transaccionales. El AWS SDK admite la interacción con la API de administración de recursos de QLDB. Para obtener más información sobre estas API, consulte Referencia de la API de Amazon QLDB.

El controlador proporciona soporte para QLDB en Java, .NET, Go, Node.js y Python. Para comenzar rápidamente a trabajar con estos lenguajes, consulte Introducción al controlador Amazon QLDB.

Antes de poder utilizar un controlador de QLDB o AWS un SDK en su aplicación, debe conceder acceso mediante programación. Para obtener más información, consulte Concesión de acceso programático.

Introducción a la consola de Amazon QLDB

Este tutorial lo guiará por los pasos necesarios para crear su primer libro mayor de Amazon QLDB y rellenarlo con tablas y datos de ejemplo. El libro mayor que crea en este ejemplo es una base de datos del departamento de vehículos automóviles (DMV) que rastrea la información histórica completa de las matriculaciones de vehículos.

El historial de un activo es un caso de uso común para QLDB porque implica una variedad de escenarios y operaciones que destacan la utilidad de una base de datos de libro mayor. Con QLDB, puede acceder, consultar y verificar directamente el historial completo de cambios en sus datos en una base de datos orientada a documentos que admite capacidades de consulta similares a las de SQL.

A lo largo de este tutorial, los siguientes temas explican cómo agregar registros de vehículos, modificarlos y ver el historial de cambios en esos registros. En esta guía también se muestra cómo verificar criptográficamente un documento de registro y, por último, se limpian los recursos y se elimina el ejemplo del libro mayor.

Cada paso del tutorial contiene instrucciones para usar AWS Management Console.

Temas

Requisitos y consideraciones previos del tutorial

Antes de empezar este tutorial de Amazon QLDB, asegúrese de que se cumplen los siguientes requisitos previos:

1. Si aún no lo ha hecho, siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Estos pasos incluyen registrarse en AWS y crear un usuario administrativo.
2. Siga las instrucciones de Configuración de permisos para configurar permisos de IAM para sus recursos de QLDB. Para completar todos los pasos de este tutorial, necesitará acceso administrativo completo a su recurso de libro mayor a través de AWS Management Console.

   > **Note**
   >
   > Si ya ha iniciado sesión como usuario con todos los permisos administrativos de AWS, puede omitir este paso.

3. (Opcional) QLDB cifra los datos en reposo mediante una clave en AWS Key Management Service (AWS KMS). Puede elijir uno de los siguientes tipos de AWS KMS keys:

   - Clave KMS propiedad de AWS: utilice una clave KMS propiedad de AWS, que también la administra en su nombre. Esta es la opción predeterminada y no requiere ninguna configuración adicional.
   - Clave KMS administrada por el cliente: utilice la clave KMS de cifrado simétrico en la cuenta que ha creado, que posee y que administra. QLDB no es compatible con claves asimétricas.

     Esta opción requiere que cree una clave KMS o use una clave existente en su cuenta. Para obtener instrucciones sobre cómo crear una clave administrada, consulte Creación de claves KMS de cifrado simétrica en la Guía para desarrolladores de AWS Key Management Service.

     Puede especificar una clave KMS administrada por el cliente utilizando un ID, un alias o el Nombre de recurso de Amazon (ARN). Para obtener más información, consulte Identificadores de clave (KeyId) en la Guía para desarrolladores de AWS Key Management Service.

     > **Note**
     >
     > No se admiten claves entre regiones. La clave KMS especificada debe estar en la misma Región de AWS que el libro mayor.

Configuración de permisos

En este paso, configurará los permisos de acceso total a través de la consola para todos los recursos de QLDB en su Cuenta de AWS. Para conceder estos permisos rápidamente, utilice la política administrada de AWS AmazonQLDBConsoleFullAccess.

Para proporcionar acceso, agregue permisos a sus usuarios, grupos o roles:

- Usuarios y grupos de AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Create a permission set (Creación de un conjunto de permisos) en la Guía del usuario de AWS IAM Identity Center.

- Usuarios administrados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones de Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda asumir. Siga las instrucciones de Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones de Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

> **Important**
>
> Para los fines de este tutorial, usted se concede acceso administrativo completo a todos los recursos de QLDB. Sin embargo, para los casos de uso de producción, siga las mejores prácticas de seguridad de concesión de privilegios mínimos o garantizando solo los permisos necesarios para realizar una tarea. Para ver ejemplos, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Para crear un libro mayor denominado `vehicle-registration`, continúe con Paso 1: crear un nuevo libro mayor.

Paso 1: crear un nuevo libro mayor

En este paso, creará un nuevo libro de mayor de Amazon QLDB denominado `vehicle-registration`. A continuación, confirma que el estado del libro mayor es Activo. También puede verificar cualquier etiqueta que haya agregado al libro mayor.

Al crear un libro mayor, se habilita de forma predeterminada la protección contra la eliminación. Se trata de una característica de QLDB que impide que los libros mayores sean eliminados por cualquier usuario. Puede deshabilitar la protección contra la eliminación al crear un libro mayor mediante la API de QLDB o AWS Command Line Interface (AWS CLI).

Para crear un nuevo libro mayor

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Introducción.
3. En la tarjeta Cree su primer libro mayor, seleccione Crear libro mayor.
4. En la página Crear libro mayor, proceda del modo siguiente:

   - Información del libro mayor: el Nombre del libro mayor debe estar rellenado previamente con **`vehicle-registration`**.
   - Modo de permisos: el modo de permisos que se va a asignar al libro mayor. Elija una de las siguientes opciones:
     - Permitir todo: un modo de permisos heredado que habilita el control de acceso con granularidad de la API para los libros mayores.

       Este modo permite a los usuarios que tengan el permiso de API `SendCommand` para este libro mayor poner en marcha todos los comandos de PartiQL (por lo tanto, `ALLOW_ALL`) en cualquier tabla del libro mayor especificado. Este modo no tendrá en cuenta las políticas de permisos de IAM de la tabla o comando que cree para el libro mayor.
     - Estándar: (recomendado) un modo de permisos que habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL. Recomendamos encarecidamente usar este modo de permisos para maximizar la seguridad de los datos del libro mayor.

       De forma predeterminada, este modo deniega todas las solicitudes para poner en marcha cualquier comando de PartiQL en tablas de este libro mayor. Para permitir los comandos

de PartiQL, debe crear políticas de permisos de IAM para recursos de tablas y acciones de PartiQL específicos, además del permiso de API `SendCommand` para el libro mayor. Para obtener información, consulte Introducción al modo de permisos estándar en Amazon QLDB.

- Cifrar datos en reposo: la clave de AWS Key Management Service (AWS KMS) que se utiliza para el cifrado de datos en reposo. Elija una de las siguientes opciones:
  - Utilice una clave KMS propiedad de AWS: utilice una clave KMS que posea y administre AWS en su nombre. Esta es la opción predeterminada y no requiere ninguna configuración adicional.
  - Elegir una clave AWS KMS diferente: utilice una clave KMS de cifrado simétrico en la cuenta que ha creado, que posee y administra.

    Para crear una clave nueva con la consola de AWS KMS, elija Crear una clave de AWS KMS. Para obtener más información, consulte Creación de claves KMS de cifrado simétricas en la Guía para desarrolladores de AWS Key Management Service.

    Para usar una clave de KMS existente, elija una de la lista desplegable o especifique un ARN de clave de KMS.
- Etiquetas: (opcional) agregue metadatos al libro mayor asociando etiquetas como pares de clave-valor. Puede añadir etiquetas a su libro mayor para ayudar en su organización e identificación. Para obtener más información, consulte Etiquetado de recursos de Amazon QLDB.

  Seleccione Añadir etiqueta y, a continuación, introduzca los pares clave-valor que desee.

5. Cuando la configuración sea la deseada, elija Crear libro mayor.

> **Note**
>
> Puede acceder a su libro mayor de QLDB cuando su estado pase a ser Activo. Esto puede tardar varios minutos.

6. En la lista de libros de mayores, busque `vehicle-registration` y confirme que el estado del libro mayor es Activo.
7. (Opcional) Elija el nombre del libro mayor `vehicle-registration`. En la página de detalles del libro mayor de registro del vehículo, confirme que todas las etiquetas que haya añadido al libro mayor aparecen en la tarjeta de etiquetas. También puede editar sus etiquetas del libro mayor con esta página de la consola.

Para crear tablas en el libro mayor `vehicle-registration`, continúe con Paso 2: crear tablas, índices y datos de muestra en un libro mayor.

Paso 2: crear tablas, índices y datos de muestra en un libro mayor

Cuando su libro mayor de Amazon QLDB esté activo, podrá empezar a crear tablas con datos sobre los vehículos, sus propietarios y su información de registro. Tras crear las tablas y los índices, puede cargarlos con datos.

En este paso, creará cuatro tablas en el libro mayor `vehicle-registration`:

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

También se crean los siguientes índices.

Nombre de la tabla	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicensePlateNumber`
`DriversLicense`	`PersonId`

Puede utilizar la consola de QLDB para crear automáticamente estas tablas con índices y cargarlas con datos de ejemplo. O bien, puede usar el editor de PartiQL de la consola para ejecutar manualmente cada instrucción de PartiQL paso a paso.

Opción automática

Cree tablas, índices y datos de muestra

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Introducción.
3. En la Opción automática de la tarjeta de datos de aplicación de muestra, seleccione `vehicle-registration` de la lista de libros mayores.
4. Elija Cargar datos de muestra.

   Si la operación finaliza correctamente, la consola muestra el mensaje Se cargaron los datos de muestra.

   Este script ejecuta todas las instrucciones en una sola transacción. Si alguna parte de la transacción falla, se revierten todas las instrucciones y se muestra el mensaje de error correspondiente. Puede volver a intentar la operación después de solucionar cualquier problema.

   > **Note**
   >
   > - Una posible causa del error de una transacción es intentar crear tablas duplicadas. La solicitud para cargar datos de muestra no se realizará correctamente si alguno de los siguientes nombres de tabla ya existe en el libro mayor: `VehicleRegistration`, `Vehicle`, `Person` y `DriversLicense`.
   >
   >   En su lugar, intente cargar estos datos de muestra en un libro mayor vacío.
   > - Este script ejecuta instrucciones `INSERT` parametrizadas. Por lo tanto, estas instrucciones de PartiQL se registran en los bloques de su diario con parámetros de enlace en lugar de los datos literales. Por ejemplo, es posible que vea la siguiente instrucción en un bloque de diario, donde el signo de interrogación (`?`) es un marcador de posición variable para el contenido del documento.
   >
   >   ```
   >   INSERT INTO Vehicle ?
   >   ```

Opción manual

Los documentos se insertan en `VehicleRegistration` con un campo `PrimaryOwner` vacío y en `DriversLicense` con un campo `PersonId` vacío. Más adelante, se rellenan estos campos con el `id` del documento de la tabla `Person` asignado por el sistema.

> **Tip**
>
> Como práctica recomendada, utilice este campo de metadatos del `id`documento como clave externa. Para obtener más información, consulte Consulta de los metadatos del documento.

Cree tablas, índices y datos de muestra

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Editor PartiQL.
3. Elija el libro mayor `vehicle-registration`.
4. Comience por crear cuatro tablas. QLDB admite contenido abierto y no aplica el esquema, por lo que no se definen atributos o tipos de datos.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar. Para ejecutar una instrucción, también puede utilizar el atajo de teclado Ctrl +Enter para Windows o Cmd+Return para macOS. Para obtener más atajos de teclado, consulte Atajos de teclado del editor PartiQL.

   ```
   CREATE TABLE VehicleRegistration
   ```

   Repita este paso para cada una de las siguientes.

   ```
   CREATE TABLE Vehicle
   ```

   ```
   CREATE TABLE Person
   ```

   ```
   CREATE TABLE DriversLicense
   ```

5. A continuación, cree índices que optimicen el rendimiento de las consultas para cada tabla.

> **Important**
>
> QLDB requiere un índice para buscar un documento de manera eficiente. Sin un índice, QLDB necesita escanear toda la tabla al leer los documentos. Esto puede provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad (`=` o `IN`) en un campo indexado o en un ID de documento. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

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

Repita este paso para lo siguiente.

```
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. Después de crear los índices, puede empezar a cargar datos en las tablas. En este paso, inserte documentos en la tabla `Person` con información personal sobre los propietarios de los vehículos que está rastreando el libro mayor.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

```
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. A continuación, rellene la tabla `DriversLicense` con documentos que incluyan la información del carné de conducir de cada propietario de vehículo.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

```
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. Ahora, rellene la tabla `VehicleRegistration` con los documentos de registro del vehículo. Estos documentos incluyen una estructura `Owners` anidada que almacena los propietarios principales y secundarios.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

```
INSERT INTO VehicleRegistration
<< {
    'VIN' : '1N4AL11D75C109151',
    'LicensePlateNumber' : 'LEWISR261LL',
    'State' : 'WA',
    'City' : 'Seattle',
    'PendingPenaltyTicketAmount' : 90.25,
    'ValidFromDate' : `2017-08-21T`,
    'ValidToDate' : `2020-05-11T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : 'KM8SRDHF6EU074761',
    'LicensePlateNumber' : 'CA762X',
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75,
    'ValidFromDate' : `2017-09-14T`,
    'ValidToDate' : `2020-06-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '3HGGK5G53FM761765',
    'LicensePlateNumber' : 'CD820Z',
    'State' : 'WA',
    'City' : 'Everett',
    'PendingPenaltyTicketAmount' : 442.30,
    'ValidFromDate' : `2011-03-17T`,
    'ValidToDate' : `2021-03-24T`,
    'Owners' : {
```

```
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '1HVBBAANXWH544237',
    'LicensePlateNumber' : 'LS477D',
    'State' : 'WA',
    'City' : 'Tacoma',
    'PendingPenaltyTicketAmount' : 42.20,
    'ValidFromDate' : `2011-10-26T`,
    'ValidToDate' : `2023-09-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
},
{
    'VIN' : '1C4RJFAG0FC625797',
    'LicensePlateNumber' : 'TH393F',
    'State' : 'WA',
    'City' : 'Olympia',
    'PendingPenaltyTicketAmount' : 30.45,
    'ValidFromDate' : `2013-09-02T`,
    'ValidToDate' : `2024-03-19T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': '' },
        'SecondaryOwners' : []
    }
} >>
```

9. Por último, rellene la tabla `Vehicle` con documentos que describan los vehículos que están registrados en su libro mayor.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

```
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'
} >>
```

A continuación, puede usar las instrucciones SELECT para leer los datos de las tablas del libro mayor `vehicle-registration`. Continúe en Paso 3: consultar las tablas en un libro mayor.

Paso 3: consultar las tablas en un libro mayor

Tras crear tablas en un libro mayor de Amazon QLDB y cargarlas con datos, puede ejecutar consultas para revisar los datos de registro del vehículo que acaba de insertar. QLDB emplea PartiQL como lenguaje de consulta y Amazon Ion como modelo de datos orientado a documentos.

PartiQL es un lenguaje de consulta de código abierto compatible con SQL que se ha ampliado para funcionar con Ion. PartiQL le permite insertar, consultar y administrar sus datos con operadores SQL conocidos. Amazon Ion es un superconjunto de JSON. Ion es un formato de datos de código abierto basado en documentos que le brinda la flexibilidad de almacenar y procesar datos estructurados, semiestructurados y anidados.

En este paso, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`.

> **Warning**
>
> Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. PartiQL admite este tipo de consultas porque es compatible con SQL. Sin embargo, no ejecute escaneados de tablas para casos de uso de producción en QLDB. Los escaneados de tablas pueden provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad en un campo indexado o en un ID de documento, por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Para consultar las tablas

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Editor PartiQL.
3. Elija el libro mayor `vehicle-registration`.
4. En la ventana del editor de consultas, introduzca la siguiente instrucción para consultar en la tabla `Vehicle` un número de identificación del vehículo (VIN) concreto que haya añadido al libro mayor y, a continuación, pulse Ejecutar.

Para ejecutar una instrucción, también puede utilizar el atajo de teclado Ctrl +Enter para Windows o Cmd+Return para macOS. Para obtener más atajos de teclado, consulte Atajos de teclado del editor PartiQL.

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

5. También puede escribir consultas de unión internas. Este ejemplo de consulta combina `Vehicle` con `VehicleRegistration` y devuelve la información de registro junto con los atributos del vehículo registrado para un `VIN` especificado.

   Introduzca la siguiente instrucción y, a continuación, seleccione Ejecutar.

   ```
   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
   ```

   También puede combinar las tablas `Person` y `DriversLicense` para ver los atributos relacionados con los conductores que se agregaron al libro mayor.

   Repita este paso para lo siguiente.

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

Para obtener información sobre la modificación de los documentos en las tablas del libro mayor `vehicle-registration`, consulte Paso 4: modificar los documentos de un libro mayor.

Paso 4: modificar los documentos de un libro mayor

Ahora que tiene datos con los que trabajar, puede empezar a realizar cambios en los documentos del libro mayor `vehicle-registration` de Amazon QLDB. Pensemos, por ejemplo, en el Audi A5 con VIN `1N4AL11D75C109151`. Este automóvil fue inicialmente propiedad de un conductor llamado Raul Lewis en Seattle, WA.

Supongamos que Raul vende el automóvil a un residente en Everett, WA, llamado Brent Logan. Entonces, Brent y Alexis Pena deciden casarse. Brent quiere añadir a Alexis como propietaria

secundaria en el registro. En este paso, las siguientes instrucciones del lenguaje de manipulación de datos (DML) muestran cómo realizar los cambios adecuados en el libro mayor para reflejar estos eventos.

> **Tip**
>
> Como práctica recomendada, utilice un `id` de documento asignado por el sistema como clave externa. Si bien puede definir campos que pretenden ser identificadores únicos (por ejemplo, el VIN de un vehículo), el verdadero identificador único de un documento es su `id`. Este campo se incluye en los metadatos del documento, que puede consultar en la vista confirmada (la vista de una tabla definida por el sistema).
> Para obtener más información acerca de las vistas en QLDB, consulte Conceptos clave.
> Para obtener más información sobre metadatos, consulte Consulta de los metadatos del documento.

Para modificar documentos

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Editor PartiQL.
3. Elija el libro mayor `vehicle-registration`.

   > **Note**
   >
   > Si configura el libro mayor mediante la característica Cargar datos de muestra de forma automática de la consola, avance al paso 6.

4. Si ejecutó las instrucciones `INSERT` manualmente para cargar los datos de muestra, continúe con estos pasos.

   Para registrar inicialmente a Raul como propietario de este vehículo, comience por buscar el `id` del documento asignado por el sistema en la tabla `Person`. Este campo se incluye en los metadatos del documento, que puede consultar en la vista de la tabla definida por el sistema llamada vista confirmada.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

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

El prefijo `_ql_committed_` es un prefijo reservado que indica que desea consultar la vista confirmada de la tabla `Person`. En esta vista, los datos están anidados en el campo `data` y los metadatos están anidados en el campo `metadata`.

5. Ahora, utilice este `id` en una instrucción UPDATE para modificar el documento correspondiente de la tabla `VehicleRegistration`. Introduzca la siguiente instrucción y, a continuación, seleccione Ejecutar.

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

Confirme que ha modificado el campo `Owners` emitiendo esta instrucción.

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

6. Para transferir la propiedad del vehículo a Brent en la ciudad de Everett, primero busque su `id` en la tabla `Person` con la siguiente instrucción.

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

A continuación, utilice este `id` para actualizar el `PrimaryOwner` y la `City` en la tabla `VehicleRegistration`.

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

Confirme que ha modificado los campos `PrimaryOwner` y `City` enviando esta instrucción.

```
SELECT r.Owners.PrimaryOwner, r.City
```

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

7. Para añadir a Alexis como propietaria secundaria del automóvil, busque su `Person id`.

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

Luego, inserte `id` en la lista `SecondaryOwners` con la siguiente instrucción de DML FROM-INSERT.

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

Confirme que ha modificado el campo `SecondaryOwners` enviando esta instrucción.

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

Para revisar estos cambios en el libro mayor `vehicle-registration`, consulte Paso 5: ver el historial de revisiones de un documento.

Paso 5: ver el historial de revisiones de un documento

Tras modificar los datos de registro de un vehículo con VIN `1N4AL11D75C109151`, puede consultar el historial de todos sus propietarios registrados y cualquier otro campo actualizado. Puede ver todas las revisiones del documento que insertó, actualizó y eliminó enviando una consulta al Función de historial integrado.

La función de historial devuelve las revisiones de la vista confirmada de la tabla, que incluye los datos de la aplicación y los metadatos asociados. Los metadatos muestran exactamente cuándo se realizó cada revisión, en qué orden y qué transacción la confirmó.

En este paso, consulta el historial de revisiones de un documento de la tabla `VehicleRegistration` del libro mayor `vehicle-registration`.

Para ver el historial de revisiones

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Editor PartiQL.
3. Elija el libro mayor `vehicle-registration`.
4. Para consultar el historial de un documento, comience por buscar su `id` único. Además de consultar la vista confirmada, otra forma de obtener el `id` de un documento consiste en utilizar la palabra clave BY en la vista de usuario predeterminada de la tabla. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

   En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

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

5. A continuación, puede utilizar este valor `id` para consultar la función de historial. Introduzca la siguiente instrucción y, a continuación, seleccione Ejecutar. Asegúrese de sustituir el valor `id` por su propio ID de documento, según proceda.

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

   > **Note**
   >
   > Para los fines de este tutorial, esta consulta de historial devuelve todas las revisiones del identificador del documento `ADR2LQq48kB9neZDupQrMm`. Como práctica recomendada, califique una consulta de historial con un identificador de documento y un intervalo de fechas (hora de inicio y hora de finalización).
   > En QLDB, cada consulta `SELECT` se procesa en una transacción y está sujeta a un límite de tiempo de espera de la transacción. Las consultas de historial que incluyen una hora de inicio y una hora de finalización se benefician de la calificación por intervalo de fechas. Para obtener más información, consulte Función de historial.

La función de historial devuelve documentos en el mismo esquema que la vista confirmada. En este ejemplo se proyectan los datos de registro del vehículo modificados. El resultado debería parecerse al siguiente:

VIN	Ciudad	Propietarios
"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**
>
> Es posible que la consulta del historial no siempre devuelva las revisiones de los documentos en orden secuencial.

Revise el resultado y confirme que los cambios reflejan lo que hizo en Paso 4: modificar los documentos de un libro mayor.

6. A continuación, puede inspeccionar los metadatos del documento de cada revisión. Introduzca la siguiente instrucción y, a continuación, seleccione Ejecutar. De nuevo, asegúrese de reemplazar el valor `id` por su propio identificador de documento, según corresponda.

```
SELECT VALUE h.metadata
```

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

El resultado debería parecerse al siguiente:

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"

Estos campos de metadatos proporcionan detalles sobre cuándo se modificó cada elemento y mediante qué transacción. A partir de estos datos, puede comprobar lo siguiente:

- El documento se identifica de forma única por su `id` asignado por el sistema: `ADR2LQq48kB9neZDupQrMm`. Se trata de un identificador único universal (UUID) que se representa en una cadena codificada en Base62.
- `txTime` muestra que la revisión inicial del documento (versión `0`) se creó en `2019-05-23T19:20:360d-3Z`.
- Cada transacción posterior crea una nueva revisión con el mismo `id` de documento, un número de versión incrementado y un `txId` y `txTime` actualizados.

Para verificar criptográficamente la revisión de un documento en el libro mayor `vehicle-registration`, continúe con Paso 6: Verificar un documento en un libro mayor.

Paso 6: Verificar un documento en un libro mayor

Con Amazon QLDB, puede verificar de manera eficiente la integridad de un documento del diario de su libro mayor mediante el uso de hash criptográfico con SHA-256. En este ejemplo, Alexis y Brent deciden cambiarse a un nuevo modelo cambiando el vehículo con VIN `1N4AL11D75C109151` en un concesionario de automóviles. El concesionario inicia el proceso verificando la propiedad del vehículo en la oficina de registro.

Para obtener más información sobre cómo funcionan la verificación y el hash criptográfico en QLDB, consulte Verificación de datos en Amazon QLDB.

En este paso, verificará la revisión de un documento en el libro mayor `vehicle-registration`. En primer lugar, solicita un resumen, que se devuelve como un archivo de salida y actúa como firma de todo el historial de cambios del libro mayor. A continuación, solicita una prueba de la revisión relativa a ese resumen. Con esta prueba, se verifica la integridad de la revisión si se aprueban todas las comprobaciones de validación.

Para solicitar un resumen

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Libros mayores.
3. En la lista de roles, seleccione `vehicle-registration`.
4. Seleccione Obtener resumen. El cuadro de diálogo Obtener resumen muestra los siguientes detalles del resumen:
   - Resumen: el valor hash SHA-256 del resumen que ha solicitado.
   - Dirección de la sugerencia del resumen: la última ubicación de bloque del diario incluida en el resumen que ha solicitado. Una dirección tiene los dos campos siguientes:
     - `strandId`: el identificador único de la cadena del diario que contiene el bloque.
     - `sequenceNo`: el número de índice que especifica la ubicación del bloque dentro de la cadena.
   - Libro mayor: nombre del libro mayor para el que ha solicitado un resumen.
   - Fecha: fecha y hora en que solicitó el resumen.
5. Revise la información del resumen. A continuación, elija Save. Puede conservar el nombre de archivo predeterminado o introducir un nombre nuevo.

Este paso guarda un archivo de texto sin formato con el contenido en formato Amazon Ion. El archivo tiene una extensión de nombre de archivo de `.ion.txt` y contiene toda la información resumida que aparecía en el cuadro de diálogo anterior. A continuación se muestra un extracto de ejemplo del contenido de un archivo de resumen. El orden de los campos puede variar en función del navegador.

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

6. Guarde este archivo en algún lugar al que pueda acceder más adelante. En los pasos siguientes, utilizará este archivo para comparar la revisión de un documento.

Una vez guardado un resumen del libro mayor, puede iniciar el proceso de verificación de una revisión de un documento comparándolo con ese resumen.

> **Note**
>
> En un caso de uso de producción para la verificación, se utiliza un resumen que se haya guardado previamente en lugar de realizar las dos tareas de forma consecutiva. Como práctica recomendada, solicite y guarde el resumen tan pronto como se escriba en el diario una revisión que desee verificar más adelante.

Para verificar la revisión de un documento

1. En primer lugar, consulte en el libro mayor el `id` y la `blockAddress` de la revisión del documento que desee comprobar. Estos campos se incluyen en los metadatos del documento, que puede consultar en la vista confirmada.

   El documento `id` es una cadena de identificación única asignada por el sistema. `blockAddress` es una estructura de Ion que especifica la ubicación del bloque en la que se efectuó la revisión.

   En el panel de navegación de la consola QLDB, elija Editor PartiQL.

2. Elija el libro mayor `vehicle-registration`.
3. En la ventana del editor de consultas, introduzca la siguiente instrucción, y a continuación elija Ejecutar.

```
SELECT r.metadata.id, r.blockAddress
FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = '1N4AL11D75C109151'
```

4. Copie y guarde los valores `id` y `blockAddress` que devuelve la consulta. Asegúrese de omitir las comillas dobles del campo `id`. En Amazon Ion, los tipos de datos de cadena se delimitan con comillas dobles.
5. Ahora que ha seleccionado una revisión del documento, puede iniciar el proceso de verificación.

   En el panel de navegación izquierdo, elija Verificaciones.
6. En el formulario Verificar documento, en Especifique el documento que desea verificar, introduzca los siguientes parámetros de entrada:

   - Libro mayor: seleccione `vehicle-registration`.
   - Dirección del bloque: el valor `blockAddress` devuelto por la consulta en el paso 3.
   - ID del documento: el valor `id` devuelto por la consulta en el paso 3.
7. En Especificar el resumen que se va a usar para la verificación, seleccione el resumen que guardó anteriormente; para ello, seleccione Elegir resumen. Si el archivo es válido, se rellenan automáticamente todos los campos de resumen de la consola. O bien, puede copiar y pegar manualmente los siguientes valores directamente desde el archivo de resumen:

   - Resumen: el valor `digest` del archivo de resumen.
   - Dirección del tip del resumen: el valor `digestTipAddress` del archivo de resumen.
8. Revise los parámetros de entrada del documento y del resumen y, a continuación, seleccione Verificar.

   La consola automatiza dos pasos:

   a. Solicite una prueba a QLDB para el documento especificado.
   b. Utilice la prueba devuelta por QLDB para llamar a una API del lado del cliente que verifica la revisión de su documento comparándola con el resumen proporcionado.

La consola muestra los resultados de su solicitud en la tarjeta de Resultados de la verificación. Para obtener más información, consulte Resultados de verificación.

9. Para probar la lógica de verificación, repita los pasos del 6 al 8 de Para verificar la revisión de un documento, pero cambie un solo carácter de la cadena de entrada del Resumen. Esto debería provocar que la solicitud de Verificación falle y muestre el mensaje de error correspondiente.

Si ya no necesita usar el libro mayor `vehicle-registration`, continúe con Paso 7 (opcional): limpiar recursos.

Paso 7 (opcional): limpiar recursos

Puede seguir utilizando el libro mayor `vehicle-registration`. Sin embargo, si ya no lo necesita, debe eliminarlo.

Si la protección contra eliminación está habilitada para su libro mayor, primero debe desactivarla para poder eliminar el libro mayor utilizando la API de QLDB, la AWS Command Line Interface (AWS CLI) o la consola de QLDB.

Para eliminar el libro mayor

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Libros mayores.
3. Si la protección contra eliminación está habilitada para este libro mayor, primero debe desactivarla.

   En la lista de libros mayores, seleccione `vehicle-registration` y, a continuación, elija Editar libro mayor.
4. En la página Editar libro mayor, desactive Protección contra las eliminaciones y, a continuación, seleccione Confirmar cambios.
5. En la lista de libros mayores, seleccione `vehicle-registration` y, a continuación, elija Eliminar.
6. Confirme ingresando **`delete vehicle-registration`** en el campo provisto.

Para obtener más información cómo trabajar con libros mayores en QLDB, consulte Introducción a Amazon QLDB: pasos siguientes.

Introducción a Amazon QLDB: pasos siguientes

Para obtener más información sobre el uso de Amazon QLDB, consulte los siguientes temas:

- Trabajar con datos e historial en Amazon QLDB
- Introducción al controlador Amazon QLDB
- Modelo de concurrencia de Amazon QLDB
- Exportación de datos de diarios desde Amazon QLDB
- Transmisión de datos de diarios desde Amazon QLDB
- Verificación de datos en Amazon QLDB
- Referencia de PartiQL de Amazon QLDB

Introducción al controlador Amazon QLDB

Este capítulo contiene tutoriales prácticos para ayudar a entender Amazon QLDB mediante el controlador de QLDB. El controlador se basa en el SDK de AWS, que admite la interacción con la API de QLDB.

Abstracción de la sesión QLDB

El controlador proporciona una capa de abstracción de alto nivel sobre la API de datos transaccionales (sesión de QLDB). Simplifica el proceso de ejecución de instrucciones PartiQL en los datos del libro mayor gestionando las llamadas a la API SendCommand. Estas llamadas a la API necesitan de varios parámetros que el controlador gestiona automáticamente, administrando las sesiones, transacciones y política de reintentos en caso de errores. El controlador también tiene optimizaciones de rendimiento y aplica las mejores prácticas para interactuar con QLDB.

> **Note**
>
> Para interactuar con las operaciones de la API de administración de recursos que se enumeran en la referencia de la API de Amazon QLDB, utilice el SDK de AWS directamente en lugar del controlador. La API de administración se utiliza únicamente para gestionar los recursos del libro mayor y para las operaciones de datos no transaccionales, como la exportación, el streaming y la verificación de datos.

Soporte de Amazon Ion

Además, el controlador utiliza las bibliotecas de Amazon Ion como soporte para gestionar los datos de Ion al ejecutar transacciones. Estas bibliotecas también se encargan de calcular el hash de los valores de Ion. QLDB requiere estos hashes de Ion para comprobar la integridad de las solicitudes de transacciones de datos.

Terminología de controladores

Esta herramienta se denomina controlador porque es comparable a otros controladores de bases de datos que proporcionan interfaces fáciles de usar para los desarrolladores. De forma similar, estos controladores encapsulan la lógica que convierte un conjunto estándar de comandos y funciones en llamadas específicas que requiere la API de bajo nivel del servicio.

El controlador es de código abierto en GitHub y está disponible para los siguientes lenguajes de programación:

- Controlador Java
- controlador .NET
- Controlador de Go
- Controlador de Node.js
- Controlador para Python

Para obtener información general sobre los controladores de todos los lenguajes de programación compatibles y tutoriales adicionales, consulte los siguientes temas:

- Gestión de sesiones con el controlador
- Recomendaciones de controladores
- Política de reintentos del controlador
- Errores comunes
- Tutorial de una aplicación de muestra
- Trabajar con Amazon Ion
- Obtener estadísticas de instrucciones PartiQL

Controlador Amazon QLDB para Java

Para trabajar con los datos de su libro mayor, puede conectarse a Amazon QLDB desde su aplicación java mediante un controlador proporcionado por AWS. En los siguientes temas se describe cómo empezar a usar el controlador QLDB para Java.

Temas

- Recursos de controladores
- Requisitos previos
- Configurar la región y las credenciales AWS predeterminadas
- Instalación
- Controlador Amazon QLDB para Java: tutorial de inicio rápido

- Controlador Amazon QLDB para Java: libro de recetas de referencia

Recursos de controladores

Para obtener más información sobre la funcionalidad compatible con el controlador Java, consulte los siguientes recursos:

- Referencia de API: 2.x, 1.x
- Código fuente del controlador (GitHub)
- Ejemplo de código fuente de aplicación (GitHub)
- Marco de carga de libros mayores (GitHub)
- Ejemplos de código de Amazon Ion

Requisitos previos

Antes de empezar a usar el controlador QLDB para Java, debe hacer lo siguiente:

1. Siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Estas incluyen las siguientes:
    1. Regístrese en AWS.
    2. Cree un usuario con los permisos de QLDB adecuados.
    3. Conceda acceso programático de desarrollo.
2. Configure un entorno de desarrollo Java descargando e instalando lo siguiente:
    1. Kit de desarrollo Java SE 8, como Amazon Corretto 8.
    2. (Opcional) Entorno de desarrollo integrado (IDE) Java de su elección, como Eclipse o IntelliJ.
3. Configurar el entorno de desarrollo para AWS SDK for Java Configurar la región y las credenciales AWS predeterminadas:

A continuación, puede descargar la aplicación de ejemplo completa del tutorial, o bien instalar solo el controlador en un proyecto de Java y ejecutar ejemplos de códigos cortos.

- Para instalar el controlador QLDB y el AWS SDK for Java en un proyecto existente, acceda a Instalación.

- Para configurar un proyecto y ejecutar ejemplos de códigos cortos que muestren las transacciones de datos básicas en un libro mayor, consulte Tutorial de inicio rápido.
- Para ver ejemplos más detallados de las operaciones de la API de datos y administración en la aplicación de ejemplo completa del tutorial, consulte Tutorial de Java.

Configurar la región y las credenciales AWS predeterminadas

El controlador QLDB y el AWS SDK for Java subyacente requieren que proporcione las credenciales de AWS a su aplicación en tiempo de ejecución. En los ejemplos de código de esta guía se supone que se usa un archivo de credenciales de AWS, tal como se describe en Configuración de las credenciales y la región predeterminadas en la Guía para desarrolladores de AWS SDK for Java 2.x.

Como parte de estos pasos, también debe establecer su valor predeterminado Región de AWS para determinar su punto de conexión de QLDB predeterminado. Los ejemplos de código se conectan a QLDB en su Región de AWS predeterminada. Para ver una lista completa de las regiones donde QLDB está disponible, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

A continuación se muestra un ejemplo de archivo de credenciales de AWS denominado `~/.aws/credentials`, donde el carácter de tilde (~) representa su directorio de inicio.

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

Cambie los valores de las credenciales de AWS por los valores *your_access_key_id* y *your_secret_access_key*.

Instalación

QLDB es compatible con las siguientes versiones de controlador y sus dependencias de SDK de AWS.

Versión de controlador	SDK de AWS	Estado	Fecha de lanzamiento más reciente
1.x	AWS SDK for Java 1.x	Lanzamiento de producción	20 de marzo de 2020

Versión de controlador	SDK de AWS	Estado	Fecha de lanzamiento más reciente
2.x	AWS SDK for Java 2.x	Lanzamiento de producción	4 de junio de 2021

Para instalar el controlador QLDB, se recomienda utilizar un sistema de administración de dependencias, como Gradle o Maven. Por ejemplo, añada el artefacto siguiente como dependencia en su proyecto Java.

2.x

Gradle

Agregue esta dependencia a su archivo de configuración `build.gradle`.

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

Maven

Agregue esta dependencia a su archivo de configuración `pom.xml`.

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

Este artefacto incluye automáticamente el módulo principal AWS SDK for Java 2.x, las bibliotecas de Amazon Ion y otras dependencias necesarias.

1.x

Gradle

Agregue esta dependencia a su archivo de configuración `build.gradle`.

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

Maven

Agregue esta dependencia a su archivo de configuración `pom.xml`.

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

Este artefacto incluye automáticamente el módulo principal AWS SDK for Java, las bibliotecas de Amazon Ion y otras dependencias necesarias.

> **Important**
>
> Espacio de nombres de Amazon Ion: al importar las clases de Amazon Ion a su aplicación, debe usar el paquete que se encuentra debajo del espacio de nombres `com.amazon.ion`. AWS SDK for Java depende de otro paquete de Ion en el espacio de nombres `software.amazon.ion`, pero se trata de un paquete heredado que no es compatible con el controlador QLDB.

Para ver ejemplos de códigos cortos sobre cómo ejecutar transacciones de datos básicos en un libro mayor, consulte la Referencia de libro de recetas.

Otras bibliotecas opcionales

Si lo desea, también puede agregar las bibliotecas útiles siguientes al proyecto. Estos artefactos son dependencias obligatorias en la aplicación de muestra Tutorial de Java.

1. aws-java-sdk-qldb: el módulo QLDB del AWS SDK for Java. La versión de QLDB mínima compatible actualmente es `1.11.785`.

Utilice este módulo en su aplicación para interactuar directamente con las operaciones de la API de administración que se enumeran en Referencia de la API de Amazon QLDB.

2. jackson-dataformat-ion: el módulo de formato de datos Jackson de FasterXML para Ion. La aplicación de ejemplo requiere una versión `2.10.0` o posterior.

Gradle

Agregue estas dependencias a su archivo de configuración `build.gradle`.

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

Maven

Agregue estas dependencias a su archivo de configuración `pom.xml`.

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

Controlador Amazon QLDB para Java: tutorial de inicio rápido

En este tutorial aprenderá a configurar una aplicación sencilla con la última versión del controlador Amazon QLDB para Java. En esta guía se incluyen los pasos para instalar el controlador y ejemplos de código breve de las operaciones básicas de creación, lectura, actualización y eliminación (CRUD). Para ver ejemplos más detallados que presentan estas operaciones en una aplicación de muestra completa, consulte Tutorial de Java.

Temas

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Si aún no lo ha hecho, complete el Requisitos previos para el controlador Java. Esto incluye registrarse en AWS, conceder acceso programático para el desarrollo e instalar un entorno de desarrollo integrado (IDE) de Java.
2. Cree un libro mayor denominado `quick-start`.

   Para obtener más información sobre cómo crear un libro mayor, consulte Operaciones básicas de libros mayores de Amazon QLDB o Paso 1: crear un nuevo libro mayor en Introducción a la consola.

Paso 1: Configuración del proyecto

En primer lugar, configure su proyecto de Java. Recomendamos usar el sistema de administración de dependencias de Maven para este tutorial.

> **Note**
>
> Si usa un IDE con características para automatizar estos pasos de configuración, puede pasar directamente a Paso 2: inicializar el controlador.

1. Cree una carpeta para su aplicación.

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

2. Introduzca el siguiente comando para inicializar el proyecto a partir de una plantilla de Maven. Reemplace *project-package*, *project-name y maven-template con* sus propios valores, según corresponda.

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

   Para *maven-template*, puede usar la plantilla básica de Maven: `maven-archetype-quickstart`

3. Para añadir el controlador QLDB para Java como una dependencia del proyecto, navegue hasta el archivo `pom.xml` recién creado y añada el siguiente artefacto.

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

   Este artefacto incluye automáticamente el módulo principal AWS SDK for Java 2.x, las bibliotecas de Amazon Ion y otras dependencias necesarias. El archivo `pom.xml` debe ser similar al siguiente:

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

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

4. Abra el archivo `App.java`.

   A continuación, añada gradualmente los ejemplos de código en los siguientes pasos para probar algunas operaciones básicas de CRUD. También puede saltarse el tutorial paso a paso y ejecutar la aplicación completa.

Paso 2: inicializar el controlador

Inicialice una instancia del controlador que se conecte al libro mayor denominado `quick-start`. Agregue el siguiente código al archivo `App.java`.

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

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

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

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

Paso 3: crear una tabla y un índice

Los siguientes ejemplos de código muestran cómo ejecutar las instrucciones `CREATE TABLE` y `CREATE INDEX`.

En el método `main`, agregue el siguiente código para crear una tabla con el nombre `People` y un índice para el campo `lastName` de dicha tabla. Los índices son necesarios para optimizar el rendimiento de las consultas y ayudar a limitar las excepciones de conflicto de control de concurrencia optimista (OCC).

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

Paso 4: insertar un documento

El siguiente ejemplo de código muestra cómo ejecutar a instrucción `INSERT`. QLDB es compatible con el lenguaje de consultas PartiQL (compatible con SQL) y el formato de datos Amazon Ion (superconjunto de JSON).

Añada el siguiente código para insertar un documento en la tabla `People`.

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

```
});
```

En este ejemplo se emplea un signo de interrogación (?) como marcador de posición variable para pasar la información del documento a la instrucción. Al utilizar marcadores de posición, debe pasar un valor de tipo `IonValue`.

> **Tip**
>
> Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo Ionlist (convertido explícitamente a `IonValue`) a la instrucción de la siguiente manera.
>
> ```
> // people is an IonList explicitly cast as an IonValue
> txn.execute("INSERT INTO People ?", (IonValue) people);
> ```
>
> No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (`<<...>>`) al pasar una `IonList`. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Paso 5: consulta del documento

El siguiente ejemplo de código muestra cómo ejecutar una instrucción `SELECT`.

Agregue el siguiente código para consultar un documento de la tabla `People`.

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

Paso 6: actualizar el documento

El siguiente ejemplo de código muestra cómo ejecutar a instrucción `UPDATE`.

1. Añada el siguiente código para actualizar un documento de la tabla `People` actualizando `age` a `42`.

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

2. Vuelva a consultar el documento para ver el valor actualizado.

```
// 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. Utilice Maven o su IDE para compilar y ejecutar el archivo `App.java`.

Ejecución de la aplicación completa

El siguiente ejemplo de código es la versión completa de la aplicación `App.java`. En lugar de seguir los pasos anteriores de forma individual, también puede copiar y ejecutar este ejemplo de código de principio a fin. Esta aplicación muestra algunas operaciones básicas de CRUD en el libro mayor denominado `quick-start`.

> **Note**
>
> Antes de ejecutar este código, asegúrese de no tener ya una tabla activa con el nombre `People` en el libro mayor `quick-start`.
> En la primera línea, sustituya *`project-package`* por el valor `groupId` que utilizó para el comando de Maven en Paso 1: Configuración del proyecto .

```
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
        });
    }
}
```

Utilice Maven o su IDE para compilar y ejecutar el archivo `App.java`.

Controlador Amazon QLDB para Java: libro de recetas de referencia

Esta guía de referencia muestra los casos de uso más comunes del controlador Amazon QLDB para Java. En él se proporcionan ejemplos de código Java que muestran cómo utilizar el controlador para ejecutar operaciones básicas de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés). También incluye ejemplos de código para procesar datos de Amazon Ion. Además, esta guía destaca las mejores prácticas para hacer que las transacciones sean idempotentes e implementar restricciones de exclusividad.

> **Note**
>
> Cuando proceda, algunos pasos incluyen ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Java.

Contenido

Importación del controlador

El siguiente ejemplo de código importa el controlador, el cliente de sesión de QLDB, los paquetes de Amazon Ion y otras dependencias relacionadas.

2.x

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

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

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

1.x

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

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

Instanciación del controlador

El siguiente ejemplo de código crea una instancia de controlador que se conecta a un nombre de libro mayor especificado y utiliza una lógica de reintentos especificada con un límite de reintentos personalizado.

> **Note**
>
> En este ejemplo también se crea una instancia de un objeto del sistema Amazon Ion (`IonSystem`). Necesita este objeto para procesar los datos de Ion al ejecutar algunas operaciones de datos de esta referencia. Para obtener más información, consulte Trabajar con 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();
```

Operaciones CRUD

La QLDB ejecuta operaciones de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés) como parte de una transacción.

> **Warning**
>
> Como práctica recomendada, haga que sus transacciones de escritura sean estrictamente idempotentes.

Hacer que las transacciones sean idempotentes

Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos. Una transacción es idempotente si puede ejecutarse varias veces y producir resultados idénticos cada vez.

Por ejemplo, pensemos en una transacción que inserta un documento en una tabla llamada `Person`. La transacción debe comprobar primero si el documento ya existe o no en la tabla. Sin esta comprobación, la tabla podría terminar con documentos duplicados.

Supongamos que la QLDB confirma correctamente la transacción en el lado del servidor pero el cliente agota el tiempo de espera mientras espera una respuesta. Si la transacción no es idempotente, se podría insertar el mismo documento más de una vez en caso de volver a intentarlo.

Uso de índices para evitar escanear tablas completas

También le recomendamos que ejecute instrucciones con una frase de predicado `WHERE` utilizando un operador de igualdad sobre un campo indexado o un ID de documento; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Sin esta búsqueda indexada, la QLDB necesita escanear las tablas, lo que puede provocar tiempos de espera de las transacciones o conflictos de control de concurrencia optimista (OCC).

Para obtener más información acerca de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Transacciones creadas de forma implícita

La función QldbDriver.execute acepta una función de Lambda que recibe una instancia de Executor, que se puede utilizar para ejecutar instrucciones. La instancia de `Executor` envuelve una transacción creada de forma implícita.

Puede ejecutar instrucciones dentro de la función de Lambda mediante el método `Executor.execute`. El controlador confirma implícitamente la transacción cuando vuelve la función de Lambda.

En las siguientes secciones se muestra cómo ejecutar operaciones CRUD básicas, especificar una lógica de reintento personalizada e implementar restricciones de exclusividad.

> **Note**
>
> Cuando proceda, en estas secciones se proporcionan ejemplos de código sobre el procesamiento de datos de Amazon Ion mediante la biblioteca Ion integrada y la biblioteca de mapeadores Jackson Ion. Para obtener más información, consulte Trabajar con Amazon Ion.

Contenido

- Eliminación de documentos
- Ejecutar varias instrucciones en una transacción
- Lógica de reintentos
- Implementación de restricciones de exclusividad

Creación de tablas

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

Creación de índices

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

Lectura de documentos

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

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

Uso de parámetros de consulta

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo Ion.

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

```
});
```

En el siguiente ejemplo de código se utilizan varios parámetros de consulta.

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

En el siguiente ejemplo de código, se utiliza una lista de parámetros de consulta.

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

Uso del mapeador Jackson

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

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

```
    }
});
```

Uso de parámetros de consulta

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo Ion.

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

En el siguiente ejemplo de código se utilizan varios parámetros de consulta.

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

En el siguiente ejemplo de código, se utiliza una lista de parámetros de consulta.

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

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

> **Note**
>
> Cuando ejecuta una consulta sin una búsqueda indexada, se invoca un escaneo completo de la tabla. En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las consultas pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Inserción de documentos

El siguiente ejemplo de código inserta los tipos de datos de Ion.

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

Uso del mapeador Jackson

El siguiente ejemplo de código inserta los tipos de datos de Ion.

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

Esta transacción inserta un documento en la tabla `Person`. Antes de insertar, compruebe primero si el documento ya existe en la tabla. Esta comprobación hace que la transacción sea de naturaleza idempotente. Incluso si realiza esta transacción varias veces, no provocará ningún efecto secundario no deseado.

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Insertar varios documentos en una instrucción

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo Ion list (convertido explícitamente a `IonValue`) a la instrucción de la siguiente manera.

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

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (`<<...>>`) al pasar una `IonList`. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

¿Por qué se requiere una conversión explícita?

El método TransactionExecutor.execute está sobrecargado. Acepta un número variable de argumentos `IonValue` (varargs) o un solo argumento `List<IonValue>`. En ion-java, `IonList` se implementa como `List<IonValue>`.

Java utiliza de forma predeterminada la implementación de método más específica cuando se llama a un método sobrecargado. En este caso, al pasar un parámetro `IonList`, el valor predeterminado es el método que toma un `List<IonValue>`. Cuando se invoca, la implementación de este método pasa los elementos `IonValue` de la lista como valores distintos. Por lo tanto, para invocar el método varargs en su lugar, debe convertir explícitamente un parámetro `IonList` a `IonValue`.

Actualización de documentos

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

Uso del mapeador Jackson

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

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Eliminación de documentos

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

Uso del mapeador Jackson

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

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Ejecutar varias instrucciones en una transacción

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

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

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

Lógica de reintentos

El método del controlador `execute` tiene un mecanismo de reintento integrado que reintenta la transacción si se produce una excepción que se pueda volver a intentar (como tiempos de espera o conflictos de OCC).

2.x

El número máximo de reintentos y la estrategia de retardo se pueden configurar.

El límite de reintentos predeterminado es 4 y la estrategia de retardo predeterminada es DefaultQldbTransactionBackoffStrategy. Puede establecer la configuración de reintentos por instancia de controlador y también por transacción mediante una instancia de RetryPolicy.

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una instancia del controlador.

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

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una transacción particular. Esta configuración `execute` anula la lógica de reintentos establecida para la instancia del controlador.

```
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

El número máximo de reintentos se puede configurar. Puede configurar el límite de reintentos estableciendo la propiedad `retryLimit` al inicializar `PooledQldbDriver`.

El límite de reintentos predeterminado es de `4`.

Implementación de restricciones de exclusividad

QLDB no admite índices únicos, pero puede implementar este comportamiento en su aplicación.

Suponga que desea implementar una restricción de exclusividad en el campo `GovId` de la tabla `Person`. Para ello, puede escribir una transacción que haga lo siguiente:

1. Afirmar que en la tabla no hay documentos existentes con un `GovId` especificado.
2. Insertar el documento si se aprueba la afirmación.

Si una transacción competidora supera la afirmación simultáneamente, solo una de las transacciones se confirmará correctamente. La otra transacción fallará y se producirá una excepción de conflicto de OCC.

En el siguiente ejemplo de código, se muestra cómo implementar esta lógica de restricción de exclusividad.

```
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**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Trabajar con Amazon Ion

En QLDB, los datos de Amazon Ion se pueden procesar de varias formas. Puede utilizar los métodos integrados de la biblioteca Ion para crear y modificar documentos de forma flexible según sea necesario. O bien, puede utilizar el módulo de formato de datos Jackson de FasterXML para Ion a fin de mapear los documentos de Ion a modelos de objetos Java simples y antiguos (POJO).

En las siguientes secciones se proporcionan ejemplos de código del procesamiento de datos de Ion mediante ambas técnicas.

Contenido

- Creación de objetos de Ion
- Lectura de objetos de Ion

Importación de los paquetes de Ion

Añada el artefacto ion-java como una dependencia en su proyecto de Java.

Gradle

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

Maven

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

Importe los siguientes paquetes Ion.

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

Uso del mapeador Jackson

Añada el artefacto jackson-dataformat-ion como una dependencia en su proyecto de Java. La QLDB requiere una versión `2.10.0` o posterior.

Gradle

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

Maven

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

Importe los siguientes paquetes Ion.

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

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

Inicialización de Ion

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

Uso del mapeador Jackson

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

Creación de objetos de Ion

El siguiente ejemplo de código crea un objeto de Ion mediante la interfaz `IonStruct` y sus métodos integrados.

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

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

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

Uso del mapeador Jackson

Suponga que tiene una clase de modelo mapeada en JSON llamada `Person`, de la siguiente manera.

```
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;
    }
}
```

El siguiente ejemplo de código crea un objeto `IonStruct` a partir de una instancia de `Person`.

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

Lectura de objetos de Ion

El siguiente ejemplo de código imprime cada campo de la instancia `ionStruct`.

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

Uso del mapeador Jackson

El siguiente ejemplo de código lee un objeto `IonStruct` y lo asigna a una instancia de `Person`.

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

Para obtener más información acerca de cómo trabajar con Ion, consulte la documentación de Amazon Ion en GitHub. Para ver más ejemplos de código sobre cómo trabajar con Ion en QLDB, consulte Uso de tipos de datos de Amazon Ion en Amazon QLDB.

Controlador Amazon QLDB para .NET

Para trabajar con los datos de su libro mayor, puede conectarse a Amazon QLDB desde su aplicación Microsoft .NET mediante un controlador proporcionado por AWS. El controlador está orientado a .NET Standard 2.0. Más específicamente, es compatible con .NET Core (LTS) 2.1+ y .NET Framework 4.5.2+. Para obtener más información sobre la compatibilidad, consulte .NET Standard en el sitio de Microsoft Docs.

Recomendamos encarecidamente usar el mapeador de objetos de Ion para omitir por completo la necesidad de conversiones manuales entre los tipos de Amazon Ion y los tipos nativos de C#.

En los siguientes temas se describe cómo empezar a usar el controlador QLDB para .NET.

Temas

- Recursos de controladores
- Requisitos previos
- Instalación
- Controlador Amazon QLDB para .NET: tutorial de inicio rápido
- Controlador Amazon QLDB para .NET: libro de recetas de referencia

Recursos de controladores

Para obtener más información sobre la funcionalidad compatible con el controlador .Net, consulte los siguientes recursos:

- Referencia de la API
- Código fuente del controlador (GitHub)
- Ejemplo de código fuente de aplicación (GitHub)
- Libro de recetas de Amazon Ion
- Mapeador de objetos de Ion (GitHub)

Requisitos previos

Antes de empezar a usar el controlador QLDB para .NET, debe hacer lo siguiente:

1. Siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Estas incluyen las siguientes:
    1. Regístrese en AWS.
    2. Cree un usuario con los permisos de QLDB adecuados.
    3. Conceda acceso programático de desarrollo.
2. Descargue e instale la versión 2.1 o posterior del SDK de .NET Core desde el sitio de descargas de Microsoft.NET.
3. (Opcional) Instale el entorno de desarrollo integrado (IDE) de su elección, como Visual Studio, Visual Studio para Mac o Visual Studio Code. Puede descargarlos desde el sitio de Microsoft Visual Studio.
4. Configurar el entorno de desarrollo para AWS SDK for .NET:
    1. Configuración de sus credenciales de AWS Recomendamos crear un archivo de credenciales compartidas.

        Para obtener más instrucciones, consulte Configurar credenciales de AWS mediante un archivo de credenciales en la Guía para desarrolladores de AWS SDK for .NET.
    2. Defina su Región de AWS predeterminada. Para saber cómo hacerlo, consulte Selección de Región de AWS.

        Para obtener una lista de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en la Referencia general de AWS.

A continuación, puede configurar una aplicación de ejemplo básica y ejecutar ejemplos de código corto, o bien puede instalar el controlador en un proyecto de .NET existente.

- Para instalar el controlador QLDB y el AWS SDK for .NET en un proyecto existente, proceda con Instalación.
- Para configurar un proyecto y ejecutar ejemplos de códigos cortos que muestren transacciones de datos básicas en un libro mayor, consulte Tutorial de inicio rápido.

Instalación

Use el administrador de paquetes NuGet para instalar el controlador QLDB para .NET. Recomendamos usar Visual Studio o el IDE de su elección para agregar las dependencias del proyecto. El nombre del paquete de controlador es Amazon.QLDB.Driver.

Por ejemplo, en Visual Studio, abra la Consola del administrador de paquetes NuGet en el menú Herramientas. A continuación, ingrese el siguiente comando en el símbolo del sistema `PM>`.

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

Al instalar el controlador también se instalan sus dependencias, incluidos AWS SDK for .NET y los paquetes de Amazon Ion.

Instale el mapeador de objetos de Ion

La versión 1.3.0 del controlador QLDB para .NET permite aceptar y devolver tipos de datos nativos de C# sin necesidad de trabajar con Amazon Ion. Para usar esta característica, añada el siguiente paquete a su proyecto.

- Amazon.QLDB.Driver.Serialization: una biblioteca que puede mapear valores de Ion a objetos CLR antiguos simples (POCO) de C# y viceversa. Este mapeador de objetos de Ion permite que su aplicación interactúe directamente con los tipos de datos nativos de C# sin que tenga que trabajar con Ion. Para consultar una breve guía sobre cómo usar esta biblioteca, acceda al archivo SERIALIZATION.md en el repositorio `awslabs/amazon-qldb-driver-dotnet` de GitHub.

Para instalar este paquete, introduzca el comando siguiente.

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

Para ver ejemplos de códigos cortos sobre cómo ejecutar transacciones de datos básicos en un libro de contabilidad, consulte Referencia de libro de recetas.

Controlador Amazon QLDB para .NET: tutorial de inicio rápido

En este tutorial aprenderá a configurar una aplicación sencilla con la última versión del controlador Amazon QLDB para .NET. En esta guía se incluyen los pasos para instalar el controlador y ejemplos de código breve de las operaciones básicas de creación, lectura, actualización y eliminación (CRUD).

Temas

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Si aún no lo ha hecho, complete el Requisitos previos para el controlador .NET. Deberá registrarse en AWS, conceder acceso programático de desarrollo e instalar el SDK de .NET Core.
2. Cree un libro mayor denominado `quick-start`.

   Para obtener más información sobre cómo crear un libro mayor, consulte Operaciones básicas de libros mayores de Amazon QLDB o Paso 1: crear un nuevo libro mayor en Introducción a la consola.

Paso 1: Configuración del proyecto

En primer lugar, configure su proyecto de .NET.

1. Para crear y ejecutar una aplicación de plantilla, ingrese los siguientes comandos `dotnet` en un terminal como bash, PowerShell o Command Prompt.

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

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

Esta plantilla crea una carpeta llamada `Amazon.QLDB.QuickStartGuide`. En esa carpeta, crea un proyecto con el mismo nombre y un archivo denominado `Program.cs`. El programa contiene un código que muestra el resultado `Hello World!`.

2. Use el administrador de paquetes NuGet para instalar el controlador QLDB para .NET. Recomendamos usar Visual Studio o el IDE de su elección para agregar las dependencias del proyecto. El nombre del paquete de controlador es Amazon.QLDB.Driver.

   - Por ejemplo, en Visual Studio, abra la Consola del administrador de paquetes NuGet en el menú Herramientas. A continuación, ingrese el siguiente comando en el símbolo del sistema `PM>`.

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

   - O bien, puede ingresar los siguientes comandos en su terminal.

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

   Al instalar el controlador también se instalan sus dependencias, incluidos AWS SDK for .NET y las bibliotecas de Amazon Ion.

3. Instale la biblioteca de serialización del controlador.

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

4. Abra el archivo `Program.cs`.

   A continuación, añada gradualmente los ejemplos de código en los siguientes pasos para probar algunas operaciones básicas de CRUD. También puede saltarse el tutorial paso a paso y ejecutar la aplicación completa.

> **Note**
>
> - Elegir entre API síncronas y asíncronas: el controlador proporciona API síncronas y asíncronas. Para las aplicaciones de alta demanda que gestionan múltiples solicitudes sin bloquearlas, recomendamos utilizar las API asíncronas para mejorar el rendimiento.

El controlador ofrece API sincrónicas como una comodidad adicional para las bases de código existentes que se escriben de forma sincrónica.

Este tutorial incluye ejemplos de código síncrono y asíncrono. Para obtener más información sobre las API, consulte las interfaces IQldbDriver y IAsyncQldbDriver en la documentación de la API.

- Procesamiento de los datos de Amazon Ion: en este tutorial, se proporcionan ejemplos de código sobre el procesamiento de datos de Amazon Ion mediante el mapeador de objetos Ion de forma predeterminada. QLDB introdujo el mapeador de objetos Ion en la versión 1.3.0 del controlador .NET. Cuando proceda, en este tutorial también se proporcionan ejemplos de código que utilizan la biblioteca Ion estándar como alternativa. Para obtener más información, consulte Trabajar con Amazon Ion.

Paso 2: inicializar el controlador

Inicialice una instancia del controlador que se conecte al libro mayor denominado `quick-start`. Agregue el siguiente código al archivo `Program.cs`.

Async

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

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

            public string LastName { get; set; }

            public int Age { get; set; }

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

```
        }

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

Sync

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

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

            public string LastName { get; set; }

            public int Age { get; set; }

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

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

```
                .Build();
        }
    }
}
```

Uso de la biblioteca Ion

Async

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

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

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

Sync

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

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

```
    {
        static IValueFactory valueFactory = new ValueFactory();

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

Paso 3: crear una tabla y un índice

Para el resto de este tutorial hasta el paso 6, debe añadir los siguientes ejemplos de código al ejemplo de código anterior.

En este paso, el siguiente código muestra cómo ejecutar las instrucciones `CREATE TABLE` y `CREATE INDEX`. Este código crea una tabla con el nombre `Person` y el índice para el campo `firstName` de esa tabla. Los índices son necesarios para optimizar el rendimiento de las consultas y ayudar a limitar las excepciones de conflicto de control de concurrencia optimista (OCC).

Async

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

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

Sync

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

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

Paso 4: insertar un documento

El siguiente ejemplo de código muestra cómo ejecutar a instrucción `INSERT`. QLDB es compatible con el lenguaje de consultas PartiQL (compatible con SQL) y el formato de datos Amazon Ion (superconjunto de JSON).

Añada el siguiente código para insertar un documento en la tabla `Person`.

Async

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

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

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

Sync

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

Person myPerson = new Person {
    FirstName = "John",
```

```
    LastName = "Doe",
    Age = 32
};

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

Uso de la biblioteca Ion

Async

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

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

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

Sync

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

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

driver.Execute(txn =>
```

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

Tip

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo Ion list a la instrucción de la siguiente manera.

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

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (<<...>>) al pasar una lista Ion. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Paso 5: consulta del documento

El siguiente ejemplo de código muestra cómo ejecutar una instrucción SELECT.

Agregue el siguiente código para consultar un documento de la tabla Person.

Async

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

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

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

```
}
```

Sync

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

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

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

Uso de la biblioteca Ion

Async

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

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

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

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

```
}
```

Sync

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

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

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

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

En este ejemplo se emplea un signo de interrogación (?) como marcador de posición variable para pasar la información del documento a la instrucción. Al utilizar marcadores de posición, debe pasar un valor de tipo `IonValue`.

Paso 6: actualizar el documento

El siguiente ejemplo de código muestra cómo ejecutar a instrucción `UPDATE`.

1. Añada el siguiente código para actualizar un documento de la tabla `Person` actualizando `age` a 42.

   Async

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

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

```
    await txn.Execute(myQuery);
});
```

Sync

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

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

2. Vuelva a consultar el documento para ver el valor actualizado.

Async

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

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

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

Sync

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

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

```
});

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

3. Para ejecutar la aplicación, introduzca el siguiente comando desde el directorio principal del directorio del proyecto `Amazon.QLDB.QuickStartGuide`.

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

Uso de la biblioteca Ion

1. Añada el siguiente código para actualizar un documento de la tabla `Person` actualizando `age` a 42.

   Async

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

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

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

   Sync

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

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

driver.Execute(txn =>
{
```

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

2. Vuelva a consultar el documento para ver el valor actualizado.

   Async

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

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

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

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

   Sync

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

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

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

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

```
}
```

3. Para ejecutar la aplicación, introduzca el siguiente comando desde el directorio principal del directorio del proyecto `Amazon.QLDB.QuickStartGuide`.

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

Ejecución de la aplicación completa

El siguiente ejemplo de código es la versión completa de la aplicación `Program.cs`. En lugar de seguir los pasos anteriores de forma individual, también puede copiar y ejecutar este ejemplo de código de principio a fin. Esta aplicación muestra algunas operaciones básicas de CRUD en el libro mayor denominado `quick-start`.

> **Note**
>
> Antes de ejecutar este código, asegúrese de no tener ya una tabla activa con el nombre `Person` en el libro mayor `quick-start`.

Async

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

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

            public string LastName { get; set; }

            public int Age { get; set; }

            public override string ToString()
            {
```

```
                return FirstName + ", " + LastName + ", " + Age.ToString();
            }
        }

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

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

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

            Console.WriteLine("Inserting a document");

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

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

            Console.WriteLine("Querying the table");

            // The result from driver.Execute() is buffered into memory because once
 the
```

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

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

            Console.WriteLine("Updating the document");

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

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

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

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

Sync

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

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

            public string LastName { get; set; }

            public int Age { get; set; }

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

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

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

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

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

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

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

            Console.WriteLine("Querying the table");

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

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

            Console.WriteLine("Updating the document");

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

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

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

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

        }
    }
}
```

Uso de la biblioteca Ion

Async

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

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

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

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

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

            Console.WriteLine("Inserting a document");

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

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

            Console.WriteLine("Querying the table");

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

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

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

```
                Console.WriteLine(row.GetField("lastName").StringValue);
                Console.WriteLine(row.GetField("age").IntValue);
            }

            Console.WriteLine("Updating the document");

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

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

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

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

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

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

Sync

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

namespace Amazon.QLDB.QuickStartGuide
```

```
{
    class Program
    {
        static IValueFactory valueFactory = new ValueFactory();

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

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

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

            Console.WriteLine("Inserting a document");

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

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

            Console.WriteLine("Querying the table");

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

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

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

            Console.WriteLine("Updating a document");

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

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

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

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

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

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

```
    }
}
```

Para ejecutar la aplicación completa, introduzca el siguiente comando desde el directorio principal del directorio del proyecto `Amazon.QLDB.QuickStartGuide`.

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

Controlador Amazon QLDB para .NET: libro de recetas de referencia

Esta guía de referencia muestra los casos de uso más comunes del controlador Amazon QLDB para .NET. En él se proporcionan ejemplos de código C# que muestran cómo utilizar el controlador para ejecutar operaciones básicas de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés). También incluye ejemplos de código para procesar datos de Amazon Ion. Además, esta guía destaca las mejores prácticas para hacer que las transacciones sean idempotentes e implementar restricciones de exclusividad.

> **Note**
>
> En este tema, se proporcionan ejemplos de código sobre el procesamiento de datos de Amazon Ion mediante el mapeador de objetos Ion de forma predeterminada. QLDB introdujo el mapeador de objetos Ion en la versión 1.3.0 del controlador .NET. Cuando proceda, en este tema también se proporcionan ejemplos de código que utilizan la biblioteca Ion estándar como alternativa. Para obtener más información, consulte Trabajar con Amazon Ion.

Contenido

Importación del controlador

En los ejemplos de código se utilizan las siguientes importaciones.

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

Uso de la biblioteca Ion

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

Instanciación del controlador

En el siguiente ejemplo de código, se crea una instancia del controlador que se conecta a un nombre de libro mayor especificado mediante la configuración predeterminada.

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();
```

Uso de la biblioteca Ion

Async

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

Sync

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

Operaciones CRUD

La QLDB ejecuta operaciones de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés) como parte de una transacción.

> **Warning**
>
> Como práctica recomendada, haga que sus transacciones de escritura sean estrictamente idempotentes.

Hacer que las transacciones sean idempotentes

Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos. Una transacción es idempotente si puede ejecutarse varias veces y producir resultados idénticos cada vez.

Por ejemplo, pensemos en una transacción que inserta un documento en una tabla llamada `Person`. La transacción debe comprobar primero si el documento ya existe o no en la tabla. Sin esta comprobación, la tabla podría terminar con documentos duplicados.

Supongamos que QLDB confirma correctamente la transacción en el lado del servidor pero el cliente agota el tiempo de espera mientras espera una respuesta. Si la transacción no es idempotente, se podría insertar el mismo documento más de una vez en caso de volver a intentarlo.

Uso de índices para evitar escanear tablas completas

También le recomendamos que ejecute instrucciones con una frase de predicado `WHERE` utilizando un operador de igualdad sobre un campo indexado o un ID de documento; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Sin esta búsqueda indexada, la QLDB necesita escanear las tablas, lo que puede provocar tiempos de espera de las transacciones o conflictos de control de concurrencia optimista (OCC).

Para obtener más información acerca de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Transacciones creadas de forma implícita

El método Amazon.QLDB.Driver.IQldbDriver.Execute acepta una función de Lambda que recibe una instancia de Amazon.QLDB.Driver.TransactionExecutor, que se puede utilizar para ejecutar instrucciones. La instancia de `TransactionExecutor` envuelve una transacción creada de forma implícita.

Puede ejecutar sentencias dentro de la función de Lambda mediante el método `Execute` del ejecutor de transacciones. El controlador confirma implícitamente la transacción cuando vuelve la función de Lambda.

En las siguientes secciones se muestra cómo ejecutar operaciones CRUD básicas, especificar una lógica de reintento personalizada e implementar restricciones de exclusividad.

Contenido

  - Insertar varios documentos en una instrucción
- Actualización de documentos
- Eliminación de documentos
- Ejecutar varias instrucciones en una transacción
- Lógica de reintentos
- Implementación de restricciones de exclusividad

Creación de tablas

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 }
}
```

Uso de la biblioteca Ion

Async

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("CREATE TABLE Person");
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Sync

```
// The result from driver.Execute() is buffered into memory because once the
// transaction is committed, streaming the result is no longer possible.
IResult result = driver.Execute(txn =>
{
    return txn.Execute("CREATE TABLE Person");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the created table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Creación de índices

Async

```
IAsyncResult<Table> createResult = await driver.Execute(async txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE INDEX ON Person(firstName)");
    return await txn.Execute(query);
});

await foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the updated table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Sync

```
IResult<Table> createResult = driver.Execute(txn =>
{
    IQuery<Table> query = txn.Query<Table>("CREATE INDEX ON Person(firstName)");
    return txn.Execute(query);
});

foreach (var result in createResult)
{
    Console.WriteLine("{ tableId: " + result.TableId + " }");
    // The statement returns the updated table ID:
    // { tableId: 4o5Uk09OcjC6PpJpLahceE }
}
```

Uso de la biblioteca Ion

Async

```
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("CREATE INDEX ON Person(GovId)");
});

await foreach (IIonValue row in result)
```

```
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Sync

```
IResult result = driver.Execute(txn =>
{
    return txn.Execute("CREATE INDEX ON Person(GovId)");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated table ID:
    // {
    //    tableId: "4o5Uk09OcjC6PpJpLahceE"
    // }
}
```

Lectura de documentos

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// Person class is defined as follows:
// public class Person
// {
//     public string GovId { get; set; }
//     public string FirstName { get; set; }
//  }

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'"));
});

await foreach (Person person in result)
```

```
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

> **Note**
>
> Cuando ejecuta una consulta sin una búsqueda indexada, se invoca un escaneo completo de la tabla. En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las consultas pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Uso de parámetros de consulta

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo C#.

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE FirstName
 = ?", "Brent"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
}
```

En el siguiente ejemplo de código, se utilizan varios parámetros de consulta de tipo C#.

```
IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId = ?
 AND FirstName = ?", "TOYENC486FH", "Brent"));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.GovId); // Prints TOYENC486FH.
    Console.WriteLine(person.FirstName); // Prints Brent.
```

```
}
```

En el siguiente ejemplo de código, se utiliza una matriz de parámetros de consulta de tipo C#.

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

string[] ids = {
    "TOYENC486FH",
    "ROEE1C1AABH",
    "YH844DA7LDB"
};

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE GovId IN
 (?,?,?)", ids));
});

await foreach (Person person in result)
{
    Console.WriteLine(person.FirstName); // Prints Brent on first iteration.
    // Prints Jim on second iteration.
    // Prints Mary on third iteration.
}
```

En el siguiente ejemplo de código, se utiliza una lista de C# como valor.

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }
// Person class is defined as follows:
// public class Person
// {
```

```
//     public string GovId { get; set; }
//     public string FirstName { get; set; }
//     public List<Vehicle> Vehicles { get; set; }
// }
// Vehicle class is defined as follows:
// public class Vehicle
// {
//     public string Make { get; set; }
//     public string Model { get; set; }
// }

List<Vehicle> vehicles = new List<Vehicle>
{
    new Vehicle
    {
        Make = "Volkswagen",
        Model = "Golf"
    },
    new Vehicle
    {
        Make = "Honda",
        Model = "Civic"
    }
};

IAsyncResult<Person> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Person>("SELECT * FROM Person WHERE Vehicles
 = ?", vehicles));
});

await foreach (Person person in result)
{
    Console.WriteLine("{");
    Console.WriteLine($"  GovId: {person.GovId},");
    Console.WriteLine($"  FirstName: {person.FirstName},");
    Console.WriteLine("  Vehicles: [");
    foreach (Vehicle vehicle in person.Vehicles)
    {
        Console.WriteLine("  {");
        Console.WriteLine($"    Make: {vehicle.Make},");
        Console.WriteLine($"    Model: {vehicle.Model},");
        Console.WriteLine("  },");
    }
```

```
    Console.WriteLine("  ]");
    Console.WriteLine("}");
    // Prints:
    // {
    //     GovId: TOYENC486FH,
    //     FirstName: Brent,
    //     Vehicles: [
    //     {
    //         Make: Volkswagen,
    //         Model: Golf
    //     },
    //     {
    //         Make: Honda,
    //         Model: Civic
    //     },
    //     ]
    // }
}
```

Uso de la biblioteca Ion

Async

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

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

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

IResult result = driver.Execute(txn =>
```

```
{
    return txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'");
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

> **Note**
>
> Cuando ejecuta una consulta sin una búsqueda indexada, se invoca un escaneo completo de la tabla. En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las consultas pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo Ion.

Async

```
IValueFactory valueFactory = new ValueFactory();
IIonValue ionFirstName = valueFactory.NewString("Brent");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE FirstName = ?",
 ionFirstName);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
IValueFactory valueFactory = new ValueFactory();
IIonValue ionFirstName = valueFactory.NewString("Brent");
```

```
IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE FirstName = ?", ionFirstName);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

En el siguiente ejemplo de código se utilizan varios parámetros de consulta.

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("Brent");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName
 = ?", ionGovId, ionFirstName);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("Brent");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName = ?",
 ionGovId, ionFirstName);
});
```

```
foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("GovId").StringValue); // Prints TOYENC486FH.
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent.
}
```

En el siguiente ejemplo de código, se utiliza una lista de parámetros de consulta.

Async

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }

IIonValue[] ionIds = {
    valueFactory.NewString("TOYENC486FH"),
    valueFactory.NewString("ROEE1C1AABH"),
    valueFactory.NewString("YH844DA7LDB")
};

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", ionIds);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent on
 first iteration.
                                                              // Prints Jim on
 second iteration.
                                                              // Prints Mary on
 third iteration.
}
```

Sync

```
// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName" : "Brent" }
// { "GovId": "ROEE1C1AABH", "FirstName" : "Jim" }
// { "GovId": "YH844DA7LDB", "FirstName" : "Mary" }
```

```
IIonValue[] ionIds = {
    valueFactory.NewString("TOYENC486FH"),
    valueFactory.NewString("ROEE1C1AABH"),
    valueFactory.NewString("YH844DA7LDB")
};

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", ionIds);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.GetField("FirstName").StringValue); // Prints Brent on
 first iteration.
                                                              // Prints Jim on
 second iteration.
                                                              // Prints Mary on
 third iteration.
}
```

En el siguiente ejemplo de código, se utiliza una lista de Ion como valor. Para obtener más información sobre los distintos tipos Ion, consulte Uso de tipos de datos de Amazon Ion en Amazon QLDB.

Async

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }

IIonValue ionVehicle1 = valueFactory.NewEmptyStruct();
ionVehicle1.SetField("Make", valueFactory.NewString("Volkswagen"));
ionVehicle1.SetField("Model", valueFactory.NewString("Golf"));
```

```
IIonValue ionVehicle2 = valueFactory.NewEmptyStruct();
ionVehicle2.SetField("Make", valueFactory.NewString("Honda"));
ionVehicle2.SetField("Model", valueFactory.NewString("Civic"));

IIonValue ionVehicles =  valueFactory.NewEmptyList();
ionVehicles.Add(ionVehicle1);
ionVehicles.Add(ionVehicle2);

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM Person WHERE Vehicles = ?",
 ionVehicles);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // Prints:
    // {
    //     GovId: "TOYENC486FN",
    //     FirstName: "Brent",
    //     Vehicles: [
    //     {
    //         Make: "Volkswagen",
    //         Model: "Golf"
    //     },
    //     {
    //         Make: "Honda",
    //         Model: "Civic"
    //     }
    //     ]
    // }
}
```

Sync

```
// Assumes that Person table has document as follows:
// { "GovId": "TOYENC486FH",
//   "FirstName" : "Brent",
//   "Vehicles": [
//      { "Make": "Volkswagen",
//        "Model": "Golf"},
```

```
//      { "Make": "Honda",
//        "Model": "Civic"}
//   ]
// }

IIonValue ionVehicle1 = valueFactory.NewEmptyStruct();
ionVehicle1.SetField("Make", valueFactory.NewString("Volkswagen"));
ionVehicle1.SetField("Model", valueFactory.NewString("Golf"));

IIonValue ionVehicle2 = valueFactory.NewEmptyStruct();
ionVehicle2.SetField("Make", valueFactory.NewString("Honda"));
ionVehicle2.SetField("Model", valueFactory.NewString("Civic"));

IIonValue ionVehicles =  valueFactory.NewEmptyList();
ionVehicles.Add(ionVehicle1);
ionVehicles.Add(ionVehicle2);

IResult result = driver.Execute(txn =>
{
    return txn.Execute("SELECT * FROM Person WHERE Vehicles = ?", ionVehicles);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // Prints:
    // {
    //     GovId: "TOYENC486FN",
    //     FirstName: "Brent",
    //     Vehicles: [
    //     {
    //         Make: "Volkswagen",
    //         Model: "Golf"
    //     },
    //     {
    //         Make: "Honda",
    //         Model: "Civic"
    //     }
    //     ]
    // }
}
```

Inserción de documentos

El siguiente ejemplo de código inserta los tipos de datos de Ion.

```
string govId = "TOYENC486FH";

Person person = new Person
{
    GovId = "TOYENC486FH",
    FirstName = "Brent"
};

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult<Person> result = await txn.Execute(txn.Query<Person>("SELECT * FROM
 Person WHERE GovId = ?", govId));

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute(txn.Query<Document>("INSERT INTO Person ?", person));
});
```

Uso de la biblioteca Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

await driver.Execute(async txn =>
{
```

```
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult result = await txn.Execute("SELECT * FROM Person WHERE GovId = ?",
 ionGovId);

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

driver.Execute(txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IResult result = txn.Execute("SELECT * FROM Person WHERE GovId = ?", ionGovId);

    // Check if there is a record in the cursor.
    int count = result.Count();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Esta transacción inserta un documento en la tabla `Person`. Antes de insertar, compruebe primero si el documento ya existe en la tabla. Esta comprobación hace que la transacción sea de naturaleza idempotente. Incluso si realiza esta transacción varias veces, no provocará ningún efecto secundario no deseado.

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Insertar varios documentos en una instrucción

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo `List` C# a la instrucción de la siguiente manera.

```
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 }
}
```

Uso de la biblioteca Ion

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo Ion list a la instrucción de la siguiente manera.

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"
    // }
}
```

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (<<...>>) al pasar una lista Ion. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Actualización de documentos

```
string govId = "TOYENC486FH";
string firstName = "John";

IAsyncResult<Document> result = await driver.Execute(async txn =>
```

```
{
    return await txn.Execute(txn.Query<Document>("UPDATE Person SET FirstName = ? WHERE
 GovId = ?", firstName , govId));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the updated document ID:
    // { documentId: Djg30Zoltqy5M4BFsA2jSJ }
}
```

Uso de la biblioteca Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("John");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?",
 ionFirstName , ionGovId);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
IIonValue ionFirstName = valueFactory.NewString("John");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("UPDATE Person SET FirstName = ? WHERE GovId = ?",
 ionFirstName , ionGovId);
```

```
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the updated document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Eliminación de documentos

```
string govId = "TOYENC486FH";

IAsyncResult<Document> result = await driver.Execute(async txn =>
{
    return await txn.Execute(txn.Query<Document>("DELETE FROM Person WHERE GovId = ?",
 govId));
});

await foreach (Document row in result)
{
    Console.WriteLine("{ documentId: " + row.DocumentId + " }");
    // The statement returns the updated document ID:
    // { documentId: Djg30Zoltqy5M4BFsA2jSJ }
}
```

Uso de la biblioteca Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");
```

```
IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("DELETE FROM Person WHERE GovId = ?", ionGovId);
});

await foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the deleted document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IResult result = driver.Execute(txn =>
{
    return txn.Execute("DELETE FROM Person WHERE GovId = ?", ionGovId);
});

foreach (IIonValue row in result)
{
    Console.WriteLine(row.ToPrettyString());
    // The statement returns the deleted document ID:
    // {
    //     documentId: "Djg30Zoltqy5M4BFsA2jSJ"
    // }
}
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Ejecutar varias instrucciones en una transacción

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

Uso de la biblioteca Ion

Async

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    ValueFactory valueFactory = new ValueFactory();
    IIonValue ionVin = valueFactory.NewString(vin);

    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
```

```
        Amazon.QLDB.Driver.IAsyncResult result = await txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 ionVin);

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
            await txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Lógica de reintentos

Para obtener información sobre la lógica de reintentos integrada en el controlador, consulte Descripción de la política de reintentos del controlador en Amazon QLDB.

Implementación de restricciones de exclusividad

QLDB no admite índices únicos, pero puede implementar este comportamiento en su aplicación.

Suponga que desea implementar una restricción de exclusividad en el campo `GovId` de la tabla `Person`. Para ello, puede escribir una transacción que haga lo siguiente:

1. Afirme que en la tabla no hay documentos existentes con un valor especificado `GovId`.
2. Insertar el documento si se aprueba la afirmación.

Si una transacción competidora supera la afirmación simultáneamente, solo una de las transacciones se confirmará correctamente. La otra transacción fallará y se producirá una excepción de conflicto de OCC.

En el siguiente ejemplo de código, se muestra cómo implementar esta lógica de restricción de exclusividad.

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

Uso de la biblioteca Ion

Async

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

await driver.Execute(async txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IAsyncResult result = await txn.Execute("SELECT * FROM Person WHERE GovId = ?",
 ionGovId);

    // Check if there is a record in the cursor.
    int count = await result.CountAsync();
```

```
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    await txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

Sync

```
IIonValue ionGovId = valueFactory.NewString("TOYENC486FH");

IIonValue ionPerson = valueFactory.NewEmptyStruct();
ionPerson.SetField("GovId", valueFactory.NewString("TOYENC486FH"));
ionPerson.SetField("FirstName", valueFactory.NewString("Brent"));

driver.Execute(txn =>
{
    // Check if a document with GovId:TOYENC486FH exists
    // This is critical to make this transaction idempotent
    IResult result = txn.Execute("SELECT * FROM Person WHERE GovId = ?", ionGovId);

    // Check if there is a record in the cursor.
    int count = result.Count();
    if (count > 0)
    {
        // Document already exists, no need to insert
        return;
    }

    // Insert the document.
    txn.Execute("INSERT INTO Person ?", ionPerson);
});
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Trabajar con Amazon Ion

En QLDB, los datos de Amazon Ion se pueden procesar de varias formas. Puede utilizar la biblioteca de Ion para crear y modificar valores Ion. O bien, puede usar el mapeador de objetos Ion para mapear objetos CLR simples y antiguos (POCO) de C# hacia y desde valores de Ion. La versión 1.3.0 del controlador QLDB para .NET introduce la compatibilidad con el mapeador de objetos Ion.

En las siguientes secciones se proporcionan ejemplos de código del procesamiento de datos de Ion mediante ambas técnicas.

Contenido

Importación del módulo Ion

```
using Amazon.IonObjectMapper;
```

Uso de la biblioteca Ion

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

Creación de tipos de Ion

En el siguiente ejemplo de código, se muestra cómo crear valores de Ion a partir de objetos de C# mediante el mapeador de objetos de Ion.

```
// Assumes that Person class is defined as follows:
// public class Person
// {
//     public string FirstName { get; set; }
//     public int Age { get; set; }
// }

// Initialize the Ion Object Mapper
IonSerializer ionSerializer = new IonSerializer();
```

```
// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

// Serialize the C# object into stream using the Ion Object Mapper
Stream stream = ionSerializer.Serialize(person);

// Load will take in stream and return a datagram; a top level container of Ion values.
IIonValue ionDatagram = IonLoader.Default.Load(stream);

// To get the Ion value within the datagram, we call GetElementAt(0).
IIonValue ionPerson = ionDatagram.GetElementAt(0);

Console.WriteLine(ionPerson.GetField("firstName").StringValue);
Console.WriteLine(ionPerson.GetField("age").IntValue);
```

Uso de la biblioteca Ion

Los siguientes ejemplos de código muestran las dos formas de crear valores Ion mediante la biblioteca de Ion.

Uso de **`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);
```

Uso de **`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);
```

Obtener un volcado binario de Ion

```
// Initialize the Ion Object Mapper with Ion binary serialization format
IonSerializer ionSerializer = new IonSerializer(new IonSerializationOptions
{
    Format = IonSerializationFormat.BINARY
});

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

MemoryStream stream = (MemoryStream) ionSerializer.Serialize(person);
Console.WriteLine(BitConverter.ToString(stream.ToArray()));
```

Uso de la biblioteca Ion

```
// ionObject is an Ion struct
MemoryStream stream = new MemoryStream();
using (var writer = IonBinaryWriterBuilder.Build(stream))
{
    ionObject.WriteTo(writer);
    writer.Finish();
}

Console.WriteLine(BitConverter.ToString(stream.ToArray()));
```

Obtener un volcado de texto de Ion

```
// Initialize the Ion Object Mapper
IonSerializer ionSerializer = new IonSerializer(new IonSerializationOptions
{
    Format = IonSerializationFormat.TEXT
});

// The C# object to be serialized
Person person = new Person
{
    FirstName = "John",
    Age = 13
};

MemoryStream stream = (MemoryStream) ionSerializer.Serialize(person);
Console.WriteLine(System.Text.Encoding.UTF8.GetString(stream.ToArray()));
```

Uso de la biblioteca Ion

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

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

Para obtener más información acerca de cómo trabajar con Ion, consulte la documentación de Amazon Ion en GitHub. Para ver más ejemplos de código sobre cómo trabajar con Ion en QLDB, consulte Uso de tipos de datos de Amazon Ion en Amazon QLDB.

Controlador Amazon QLDB para Go

Para trabajar con los datos de su libro mayor, puede conectarse a Amazon QLDB desde su aplicación Go mediante un controlador proporcionado por AWS. En los siguientes temas se describe cómo empezar a usar el controlador QLDB para Go.

Temas

Recursos de controladores

Para obtener más información sobre las funcionalidades compatibles con el controlador Go, consulte los siguientes recursos:

- Referencia de API: 3.x, 2.x, 1.x
- Código origen del controlador (GitHub)
- Libro de recetas de Amazon Ion

Requisitos previos

Antes de empezar a usar el controlador QLDB para Go, debe hacer lo siguiente:

1. Siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Estas incluyen las siguientes:
    1. Regístrese en AWS.
    2. Cree un usuario con los permisos de QLDB adecuados.
    3. Conceda acceso programático de desarrollo.
2. (Opcional) Instale un entorno de desarrollo integrado (IDE) de su elección. Para obtener una lista de los IDE más utilizados para Go, consulte los complementos e IDE del editor en el sitio web de Go.
3. Descargue e instale una de las siguientes versiones de Go desde el sitio de descargas de Go:
    - 1.15 o posterior: controlador de QLDB para Go v3
    - 1.14: controlador de la QLDB para Go v1 o v2
4. Configure el entorno de desarrollo para AWS SDK for Go:
    1. Configure sus credenciales de AWS. Recomendamos crear un archivo de credenciales compartidas.

Para obtener instrucciones, consulte especificación de credenciales en la Guía para desarrolladores de AWS SDK for Go.

2. Defina la Región de AWS predeterminada. Para obtener información sobre cómo hacerlo, consulte Especificar la Región de AWS.

   Para ver una lista completa de las regiones disponibles, consulte puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

A continuación, puede configurar una aplicación de ejemplo básica y ejecutar ejemplos de código corto, o bien puede instalar el controlador en un proyecto de Go existente.

- Para instalar el controlador QLDB y AWS SDK for Go en un proyecto existente, acceda a Instalación.
- Para configurar un proyecto y ejecutar ejemplos de códigos cortos que muestren las transacciones de datos básicas en un libro mayor, consulte Tutorial de inicio rápido.

Instalación

El controlador de la QLDB para Go es de código abierto en el repositorio awslabs/amazon-qldb-driver-go de GitHub. QLDB es compatible con las siguientes versiones de controlador y sus dependencias de Go.

Versión de controlador	Versión de Go	Estado	Fecha de lanzamiento más reciente
1.x	1.14 o posteriores	Lanzamiento de producción	16 de junio de 2021
2.x	1.14 o posteriores	Lanzamiento de producción	21 de julio de 2021
3.x	1.15 o posteriores	Lanzamiento de producción	10 de noviembre de 2022

Para instalar el controlador

1. Asegúrese de que su proyecto utilice los módulos de Go para instalar las dependencias del proyecto.
2. Ejecute el siguiente comando `go get` en el directorio de su proyecto.

   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
   ```

Al instalar el controlador también se instalan sus dependencias, incluidos AWS SDK for Go o AWS SDK for Go v2 y los paquetes de Amazon Ion.

Para ver ejemplos de códigos cortos sobre cómo ejecutar transacciones de datos básicos en un libro mayor, consulte la Referencia de libro de recetas.

Controlador Amazon QLDB para Go: tutorial de inicio rápido

En este tutorial aprenderá a configurar una aplicación sencilla con la última versión del controlador Amazon QLDB para Go. En esta guía se incluyen los pasos para instalar el controlador y ejemplos de código breve de las operaciones básicas de creación, lectura, actualización y eliminación (CRUD).

Temas

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Si aún no lo ha hecho, complete el Requisitos previos para el controlador Go. Deberá registrarse en AWS, conceder acceso programático de desarrollo e instalar Go.
2. Cree un libro mayor denominado `quick-start`.

   Para obtener más información sobre cómo crear un libro mayor, consulte Operaciones básicas de libros mayores de Amazon QLDB o Paso 1: crear un nuevo libro mayor en Introducción a la consola.

Paso 1: instalar el controlador

Asegúrese de que su proyecto utilice los módulos de Go para instalar las dependencias del proyecto.

Ejecute el siguiente comando `go get` en el directorio de su proyecto.

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

Al instalar el controlador también se instalan sus dependencias, incluidos AWS SDK for Go v2 y los paquetes de Amazon Ion.

Paso 2: importar los paquetes

Importe los siguientes paquetes AWS.

```
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"
)
```

Paso 3: inicializar el controlador

Inicialice una instancia del controlador que se conecte al libro mayor denominado `quick-start`.

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

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

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

> **Note**
>
> En este ejemplo de código, sustituya *us-east-1* por el Región de AWS donde creó el libro mayor.

Paso 4: crear una tabla y un índice

Los siguientes ejemplos de código muestran cómo ejecutar las instrucciones `CREATE TABLE` y `CREATE INDEX`.

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

```
    if err != nil {
        return nil, err
    }

    // When working with QLDB, it's recommended to create an index on fields we're
 filtering on.
    // This reduces the chance of OCC conflict exceptions with large datasets.
    _, err = txn.Execute("CREATE INDEX ON People (firstName)")
    if err != nil {
        return nil, err
    }

    _, err = txn.Execute("CREATE INDEX ON People (age)")
    if err != nil {
        return nil, err
    }

    return nil, nil
})
if err != nil {
    panic(err)
}
```

Este código crea una tabla con el nombre `People` y los índices de los campos `firstName` y `age` de esa tabla. Los índices son necesarios para optimizar el rendimiento de las consultas y ayudar a limitar las excepciones de conflicto de control de concurrencia optimista (OCC).

Paso 5: insertar un documento

El siguiente ejemplo de código muestra cómo ejecutar la instrucción `INSERT`. QLDB es compatible con el lenguaje de consultas PartiQL (compatible con SQL) y el formato de datos Amazon Ion (superconjunto de JSON).

Uso de literales PartiQL

El siguiente código inserta un documento en la tabla `People` mediante una instrucción PartiQL de literal de cadena.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("INSERT INTO People {'firstName': 'Jane', 'lastName': 'Doe',
 'age': 77}")
```

```
})
if err != nil {
    panic(err)
}
```

Uso de tipos de datos de Ion

De forma similar al paquete JSON integrado en Go, puede serializar y anular la serialización de los tipos de datos de Go desde y hacia Ion.

1. Suponga que tiene la siguiente estructura de Go denominada `Person`.

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

2. Cree una instancia de `Person`.

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

   El controlador serializa automáticamente una representación de texto codificada en Ion de `person`.

   > **Important**
   >
   > Para que la serialización y la anulación de la serialización funcionen correctamente, se deben exportar los nombres de campo de la estructura de datos de Go (primera letra en mayúscula).

3. Pase la instancia `person` al método `Execute` de la transacción.

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

En este ejemplo se emplea un signo de interrogación (?) como marcador de posición variable para pasar la información del documento a la instrucción. Al utilizar marcadores de posición, debe pasar un valor de texto codificado por Ion.

> **Tip**
>
> Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo list a la instrucción de la siguiente manera.
>
> ```
> // people is a list
> txn.Execute("INSERT INTO People ?", people)
> ```
>
> No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (<<...>>) al pasar una lista. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Paso 6: consulta del documento

El siguiente ejemplo de código muestra cómo ejecutar una instrucción `SELECT`.

```
p, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE age =
 54")
    if err != nil {
        return nil, err
    }

    // Assume the result is not empty
    hasNext := result.Next(txn)
    if !hasNext && result.Err() != nil {
        return nil, result.Err()
    }

    ionBinary := result.GetCurrentData()

    temp := new(Person)
    err = ion.Unmarshal(ionBinary, temp)
    if err != nil {
```

```
        return nil, err
    }

    return *temp, nil
})
if err != nil {
    panic(err)
}

var returnedPerson Person
returnedPerson = p.(Person)

if returnedPerson != person {
    fmt.Print("Queried result does not match inserted struct")
}
```

En este ejemplo, se consulta el documento desde la tabla `People`, se supone que el conjunto de resultados no está vacío y se devuelve el documento a partir del resultado.

Paso 7: actualizar el documento

El siguiente ejemplo de código muestra cómo ejecutar a instrucción `UPDATE`.

```
person.Age += 10

_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    return txn.Execute("UPDATE People SET age = ? WHERE firstName = ?", person.Age,
 person.FirstName)
})
if err != nil {
    panic(err)
}
```

Paso 8: consulta al documento actualizado

El siguiente ejemplo de código consulta la tabla `People` mediante `firstName` y devuelve todos los documentos del conjunto de resultados.

```
p, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
 firstName = ?", person.FirstName)
```

```
    if err != nil {
        return nil, err
    }

    var people []Person
    for result.Next(txn) {
        ionBinary := result.GetCurrentData()

        temp := new(Person)
        err = ion.Unmarshal(ionBinary, temp)
        if err != nil {
            return nil, err
        }

        people = append(people, *temp)
    }
    if result.Err() != nil {
        return nil, result.Err()
    }

    return people, nil
})
if err != nil {
    panic(err)
}

var people []Person
people = p.([]Person)

updatedPerson := Person{"John", "Doe", 64}
if people[0] != updatedPerson {
    fmt.Print("Queried result does not match updated struct")
}
```

Paso 9: descarte de la tabla

El siguiente ejemplo de código muestra cómo ejecutar una instrucción `DROP TABLE`.

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

```
    panic(err)
}
```

Ejecución de la aplicación completa

El siguiente ejemplo de código es la versión completa de la aplicación. En lugar de seguir los pasos anteriores de forma individual, también puede copiar y ejecutar este ejemplo de código de principio a fin. Esta aplicación muestra algunas operaciones básicas de CRUD en el libro mayor denominado `quick-start`.

> **Note**
>
> Antes de ejecutar este código, asegúrese de no tener ya una tabla activa con el nombre `People` en el libro mayor `quick-start`.

```
package main

import (
    "context"
    "fmt"

    "github.com/amzn/ion-go/ion"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/qldbsession"
    "github.com/awslabs/amazon-qldb-driver-go/v2/qldbdriver"
)

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

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

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

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

        // When working with QLDB, it's recommended to create an index on fields we're
filtering on.
        // This reduces the chance of OCC conflict exceptions with large datasets.
        _, err = txn.Execute("CREATE INDEX ON People (firstName)")
        if err != nil {
            return nil, err
        }

        _, err = txn.Execute("CREATE INDEX ON People (age)")
        if err != nil {
            return nil, err
        }

        return nil, nil
    })
    if err != nil {
        panic(err)
    }

    _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        return txn.Execute("INSERT INTO People {'firstName': 'Jane', 'lastName': 'Doe',
'age': 77}")
    })
    if err != nil {
        panic(err)
    }

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

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

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

    p, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
        result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
age = 54")
        if err != nil {
            return nil, err
        }

        // Assume the result is not empty
        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return nil, result.Err()
        }

        ionBinary := result.GetCurrentData()

        temp := new(Person)
        err = ion.Unmarshal(ionBinary, temp)
        if err != nil {
            return nil, err
        }

        return *temp, nil
    })
    if err != nil {
        panic(err)
    }

    var returnedPerson Person
    returnedPerson = p.(Person)

    if returnedPerson != person {
        fmt.Print("Queried result does not match inserted struct")
    }
```

```
   person.Age += 10

   _, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
       return txn.Execute("UPDATE People SET age = ? WHERE firstName = ?", person.Age,
person.FirstName)
   })
   if err != nil {
       panic(err)
   }

   p, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
(interface{}, error) {
       result, err := txn.Execute("SELECT firstName, lastName, age FROM People WHERE
firstName = ?", person.FirstName)
       if err != nil {
           return nil, err
       }

       var people []Person
       for result.Next(txn) {
           ionBinary := result.GetCurrentData()

           temp := new(Person)
           err = ion.Unmarshal(ionBinary, temp)
           if err != nil {
               return nil, err
           }

           people = append(people, *temp)
       }
       if result.Err() != nil {
           return nil, result.Err()
       }

       return people, nil
   })
   if err != nil {
       panic(err)
   }

   var people []Person
   people = p.([]Person)
```

```
    updatedPerson := Person{"John", "Doe", 64}
    if people[0] != updatedPerson {
        fmt.Print("Queried result does not match updated struct")
    }

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

Controlador Amazon QLDB para Go: libro de recetas de referencia

Esta guía de referencia muestra los casos de uso más comunes del controlador Amazon QLDB para Go. En él se proporcionan ejemplos de código Go que muestran cómo utilizar el controlador para ejecutar operaciones básicas de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés). También incluye ejemplos de código para procesar datos de Amazon Ion. Además, esta guía destaca las mejores prácticas para hacer que las transacciones sean idempotentes e implementar restricciones de exclusividad.

> **Note**
>
> Cuando proceda, algunos pasos incluyen ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Go.

Contenido

Importación del controlador

El siguiente ejemplo de código importa el controlador y otros paquetes AWS necesarios.

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**
>
> En este ejemplo también se importa el paquete Amazon Ion (`amzn/ion-go/ion`). Necesita este paquete para procesar los datos de Ion al ejecutar algunas operaciones de datos de esta referencia. Para obtener más información, consulte Trabajar con Amazon Ion.

Instanciación del controlador

En el siguiente ejemplo de código, se crea una instancia del controlador que se conecta a un nombre de libro mayor especificado en una Región de AWS determinada.

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)
}
```

Operaciones CRUD

La QLDB ejecuta operaciones de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés) como parte de una transacción.

> **Warning**
>
> Como práctica recomendada, haga que sus transacciones de escritura sean estrictamente idempotentes.

Hacer que las transacciones sean idempotentes

Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos. Una transacción es idempotente si puede ejecutarse varias veces y producir resultados idénticos cada vez.

Por ejemplo, pensemos en una transacción que inserta un documento en una tabla llamada `Person`. La transacción debe comprobar primero si el documento ya existe o no en la tabla. Sin esta comprobación, la tabla podría terminar con documentos duplicados.

Supongamos que la QLDB confirma correctamente la transacción en el lado del servidor pero el cliente agota el tiempo de espera mientras espera una respuesta. Si la transacción no es idempotente, se podría insertar el mismo documento más de una vez en caso de volver a intentarlo.

Uso de índices para evitar escanear tablas completas

También le recomendamos que ejecute instrucciones con una frase de predicado `WHERE` utilizando un operador de igualdad en un campo indexado o un ID de documento; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Sin esta búsqueda

indexada, la QLDB necesita escanear las tablas, lo que puede provocar tiempos de espera de las transacciones o conflictos de control de concurrencia optimista (OCC).

Para obtener más información acerca de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Transacciones creadas de forma implícita

La función QLDBDriver.Execute acepta una función de Lambda que recibe una instancia de Transaction, que se puede utilizar para ejecutar instrucciones. La instancia de `Transaction` envuelve una transacción creada de forma implícita.

Puede ejecutar instrucciones dentro de la función de Lambda mediante la función `Transaction.Execute`. El controlador confirma implícitamente la transacción cuando vuelve la función de Lambda.

En las siguientes secciones se muestra cómo ejecutar operaciones CRUD básicas, especificar una lógica de reintento personalizada e implementar restricciones de exclusividad.

Contenido

Creación de tablas

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

Creación de índices

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

Lectura de documentos

```
var decodedResult map[string]interface{}

// Assumes that Person table has documents as follows:
// { "GovId": "TOYENC486FH", "FirstName": "Brent" }
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = 'TOYENC486FH'")
  if err != nil {
    return nil, err
  }
  for result.Next(txn) {
    ionBinary := result.GetCurrentData()
    err = ion.Unmarshal(ionBinary, &decodedResult)
    if err != nil {
      return nil, err
    }
    fmt.Println(decodedResult) // prints map[GovId: TOYENC486FH FirstName:Brent]
  }
  if result.Err() != nil {
    return nil, result.Err()
  }
  return nil, nil
})
if err != nil {
  panic(err)
}
```

Uso de parámetros de consulta

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo nativo.

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

```
  return txn.Execute("SELECT * FROM Person WHERE GovId = ?", "TOYENC486FH")
})
if err != nil {
  panic(err)
}
```

En el siguiente ejemplo de código se utilizan varios parámetros de consulta.

```
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId = ? AND FirstName = ?",
 "TOYENC486FH", "Brent")
})
if err != nil {
  panic(err)
}
```

En el siguiente ejemplo de código, se utiliza una lista de parámetros de consulta.

```
govIDs := []string{}{"TOYENC486FH", "ROEE1", "YH844"}

result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  return txn.Execute("SELECT * FROM Person WHERE GovId IN (?,?,?)", govIDs...)
})
if err != nil {
  panic(err)
}
```

> **Note**
>
> Cuando ejecuta una consulta sin una búsqueda indexada, se invoca un escaneo completo de la tabla. En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las consultas pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Inserción de documentos

El siguiente ejemplo de código inserta los tipos de datos nativos.

```
_, err = driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
  // Check if a document with a GovId of TOYENC486FH exists
  // This is critical to make this transaction idempotent
  result, err := txn.Execute("SELECT * FROM Person WHERE GovId = ?", "TOYENC486FH")
  if err != nil {
    return nil, err
  }
  // Check if there are any results
  if result.Next(txn) {
    // Document already exists, no need to insert
  } else {
    person := map[string]interface{}{
      "GovId": "TOYENC486FH",
      "FirstName": "Brent",
    }
    _, err = txn.Execute("INSERT INTO Person ?", person)
    if err != nil {
      return nil, err
    }
  }
  return nil, nil
})
```

Esta transacción inserta un documento en la tabla `Person`. Antes de insertar, compruebe primero si el documento ya existe en la tabla. Esta comprobación hace que la transacción sea de naturaleza idempotente. Incluso si realiza esta transacción varias veces, no provocará ningún efecto secundario no deseado.

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Insertar varios documentos en una instrucción

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo list a la instrucción de la siguiente manera.

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

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (<<...>>) al pasar una lista. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Actualización de documentos

El siguiente ejemplo de código utiliza tipos de datos nativos.

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

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Eliminación de documentos

El siguiente ejemplo de código utiliza tipos de datos nativos.

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

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Ejecutar varias instrucciones en una transacción

```
// This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
// set your UPDATE to filter on vin and insured, and check if you updated something or
 not.
func InsureCar(driver *qldbdriver.QLDBDriver, vin string) (bool, error) {
    insured, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {

        result, err := txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
        if err != nil {
            return false, err
        }

        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return false, result.Err()
        }

        if hasNext {
            _, err = txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
            if err != nil {
                return false, err
            }
            return true, nil
        }
        return false, nil
    })
    if err != nil {
        panic(err)
    }

    return insured.(bool), err
}
```

Lógica de reintentos

La función del controlador `Execute` tiene un mecanismo de reintento integrado que reintenta la transacción si se produce una excepción que se pueda volver a intentar (como tiempos de espera o conflictos de OCC). El número máximo de reintentos y la estrategia de retardo se pueden configurar.

El límite de reintentos predeterminado es 4 y la estrategia de retardo predeterminada es ExponentialBackoffStrategy, con una base de 10 milisegundos. Puede establecer la política de reintentos por instancia de controlador y también por transacción mediante una instancia de RetryPolicy.

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una instancia del controlador.

```
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)
```

```
  }
}
```

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una función anónima particular. La función `SetRetryPolicy` anula la política de reintentos establecida para la instancia del controlador.

```
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")
  })
}
```

Implementación de restricciones de exclusividad

QLDB no admite índices únicos, pero puede implementar este comportamiento en su aplicación.

Suponga que desea implementar una restricción de exclusividad en el campo `GovId` de la tabla `Person`. Para ello, puede escribir una transacción que haga lo siguiente:

1. Afirmar que en la tabla no hay documentos existentes con un `GovId` especificado.
2. Insertar el documento si se aprueba la afirmación.

Si una transacción competidora supera la afirmación simultáneamente, solo una de las transacciones se confirmará correctamente. La otra transacción fallará y se producirá una excepción de conflicto de OCC.

En el siguiente ejemplo de código, se muestra cómo implementar esta lógica de restricción de exclusividad.

```
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**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Trabajar con Amazon Ion

Las siguientes secciones muestran cómo usar el módulo Amazon Ion para procesar los datos de Ion.

Contenido

Importación del módulo Ion

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

Creación de tipos de Ion

La biblioteca de Ion para Go actualmente no es compatible con el modelo de objetos de documento (DOM), por lo que no se pueden crear tipos de datos de Ion. Pero puede serializar y anular la serialización entre los tipos nativos de Go y binario de Ion cuando trabaja con la QLDB.

Obtener el binario de 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]
```

Obtener texto de Ion

```
aDict := map[string]interface{}{
  "GovId": "TOYENC486FH",
  "FirstName": "Brent",
}

ionBytes, err := ion.MarshalText(aDict)
if err != nil {
  panic(err)
}

fmt.Println(string(ionBytes)) // prints {FirstName:"Brent",GovId:"TOYENC486FH"}
```

Para obtener más información acerca de Ion, consulte la documentación de Amazon Ion en GitHub. Para ver más ejemplos de código sobre cómo trabajar con Ion en QLDB, consulte Uso de tipos de datos de Amazon Ion en Amazon QLDB.

Controlador Amazon QLDB para Node.js

Para trabajar con los datos de su libro de contabilidad, puede conectarse a Amazon QLDB desde su aplicación Node.js mediante un controlador proporcionado por AWS. En los siguientes temas se describe cómo empezar a usar el controlador QLDB para Node.js.

Temas

- Recursos de controladores
- Requisitos previos
- Instalación
- Recomendaciones de configuración

- Controlador Amazon QLDB para Node.js: tutorial de inicio rápido
- Controlador Amazon QLDB para Node.js: libro de recetas de referencia

Recursos de controladores

Para obtener más información sobre las funcionalidades compatibles con el controlador Node.js, consulte los siguientes recursos:

- Referencia de API: 3.x, 2.x, 1.x
- Código fuente del controlador (GitHub)
- Ejemplo de código fuente de aplicación (GitHub)
- Ejemplos de código de Amazon Ion
- Cree una operación CRUD sencilla y un flujo de datos en QLDB con AWS Lambda (Blog de AWS)

Requisitos previos

Antes de empezar a usar el controlador QLDB para Node.js, debe hacer lo siguiente:

1. Siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Esta incluye lo siguiente:
    1. Regístrese en AWS.
    2. Cree un usuario con los permisos de QLDB adecuados.
    3. Conceda acceso programático de desarrollo.
2. Instalar la versión 14.x o posterior de Node.js que encontrará en el sitio de descargas de Node.js. (Las versiones anteriores del controlador son compatibles con la versión 10.x o posterior de Node.js).
3. Configurar su entorno de desarrollo para el SDK de AWS para JavaScript en Node.js:
    1. Configuración de sus credenciales de AWS Recomendamos crear un archivo de credenciales compartidas.

        Para ver las instrucciones, consulte Carga de credenciales en Node.js desde el archivo de credenciales compartidas en la Guía para desarrolladores de AWS SDK for JavaScript.
    2. Defina la Región de AWS predeterminada. Para aprender cómo, consulte Configuración de Región de AWS.

Para obtener una lista de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en la Referencia general de AWS.

A continuación, puede descargar la aplicación de ejemplo completa del tutorial, o bien instalar solo el controlador en un proyecto de Node.js y ejecutar ejemplos de códigos cortos.

- Para instalar el controlador QLDB y el SDK de AWS para JavaScript en Node.js en un proyecto existente, acceda a Instalación.
- Para configurar un proyecto y ejecutar ejemplos de códigos cortos que muestren las transacciones de datos básicas en un libro de contabilidad, consulte Tutorial de inicio rápido.
- Para ver ejemplos más detallados de las operaciones de la API de datos y administración en la aplicación de ejemplo completa del tutorial, consulte Tutorial de Node.js.

Instalación

QLDB es compatible con las siguientes versiones de controlador y sus dependencias de Node.js.

Versión de controlador	Node.js version	Status	Fecha de lanzamiento más reciente
1.x	10.x o posterior	Lanzamiento de producción	5 de junio de 2020
2.x	10.x o posterior	Lanzamiento de producción	6 de mayo de 2021
3.x	14.x o posterior	Lanzamiento de producción	10 de noviembre de 2023

Para instalar el controlador QLDB mediante npm (el administrador de paquetes de Node.js), introduzca el siguiente comando desde el directorio raíz del proyecto.

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
```

El controlador tiene las siguientes dependencias de pares en los siguientes paquetes: También debe instalar estos paquetes como dependencias en su proyecto.

3.x

Cliente de QLDB agregado y modular (API de administración)

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

Cliente de sesión de QLDB agregado y modular (API de datos)

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

Formato de datos Amazon Ion

```
npm install ion-js
```

Implementación de JavaScript puro de `BigInt`

```
npm install jsbi
```

2.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Formato de datos Amazon Ion

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

Implementación de JavaScript puro de `BigInt`

```
npm install jsbi@3.1.1
```

1.x

AWS SDK for JavaScript

```
npm install aws-sdk
```

Formato de datos Amazon Ion

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

Implementación de JavaScript puro de `BigInt`

```
npm install jsbi@3.1.1
```

Uso del controlador para conectar a un libro de contabilidad

A continuación, puede importar el controlador y usarlo para conectar a un libro mayor. En el siguiente ejemplo de código de TypeScript se muestra cómo crear una instancia de controlador para un nombre de libro de contabilidad y Región de AWS específicos.

3.x

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

const maxConcurrentTransactions: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: new Agent({
      maxSockets: maxConcurrentTransactions
    })
};
```

```
const serviceConfigurationOptions: QLDBSessionClientConfig = {
    region: "us-east-1"
};

//Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
const retryConfig: RetryConfig = new RetryConfig(retryLimit);
const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, lowLevelClientHttpOptions, maxConcurrentTransactions,
 retryConfig);

qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

2.x

```
import { Agent } from 'https';
import { QldbDriver, RetryConfig  } from 'amazon-qldb-driver-nodejs';

const maxConcurrentTransactions: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const agentForQldb: Agent = new Agent({
    keepAlive: true,
    maxSockets: maxConcurrentTransactions
});

const serviceConfigurationOptions = {
    region: "us-east-1",
    httpOptions: {
        agent: agentForQldb
    }
};

//Use driver's default backoff function for this example (no second parameter
 provided to RetryConfig)
const retryConfig: RetryConfig = new RetryConfig(retryLimit);
const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);
```

```
qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

1.x

```
import { Agent } from 'https';
import { QldbDriver } from 'amazon-qldb-driver-nodejs';

const poolLimit: number = 10;
const retryLimit: number = 4;

//Reuse connections with keepAlive
const agentForQldb: Agent = new Agent({
    keepAlive: true,
    maxSockets: poolLimit
});

const serviceConfigurationOptions = {
    region: "us-east-1",
    httpOptions: {
        agent: agentForQldb
    }
};

const qldbDriver: QldbDriver = new QldbDriver("testLedger",
 serviceConfigurationOptions, retryLimit, poolLimit);
qldbDriver.getTableNames().then(function(tableNames: string[]) {
    console.log(tableNames);
});
```

Para ver ejemplos de códigos cortos sobre cómo ejecutar transacciones de datos básicos en un libro de contabilidad, consulte la Referencia de libro de recetas.

Recomendaciones de configuración

Reutilización de conexiones con keep-alive

Controlador QLDB Node.js v3

El agente HTTP o HTTPS predeterminado de Node.js crea una nueva conexión TCP para cada nueva solicitud. Para evitar el coste de establecer una nueva conexión, la versión 3 de AWS SDK for

JavaScript reutiliza las conexiones TCP de forma predeterminada. Para obtener más información y aprender a deshabilitar la reutilización de conexiones, consulte Reutilizar conexiones con keep-alive en Node.js en la Guía para desarrolladores de AWS SDK for JavaScript.

Recomendamos usar la configuración predeterminada para reutilizar conexiones del controlador QLDB para Node.js. Durante la inicialización del controlador, defina la opción HTTP `maxSockets` del cliente de bajo nivel en el mismo valor que estableció para `maxConcurrentTransactions`.

Para ver un ejemplo, consulte el siguiente código de JavaScript o TypeScript.

JavaScript

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

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

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

const lowLevelClientHttpOptions = {
    httpAgent: agentForQldb
}

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

TypeScript

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

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

const agentForQldb: Agent = new Agent({
```

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

const lowLevelClientHttpOptions: NodeHttpHandlerOptions = {
    httpAgent: agentForQldb
};

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

Controlador QLDB Node.js v2

El agente HTTP o HTTPS predeterminado de Node.js crea una nueva conexión TCP para cada nueva solicitud. Para evitar el costo que supone establecer una nueva conexión, recomendamos reutilizar una conexión existente.

Para reutilizar las conexiones del controlador QLDB para Node.js, use una de las siguientes opciones:

- Durante la inicialización del controlador, configure las siguientes opciones HTTP del cliente de bajo nivel:
  - `keepAlive` – `true`
  - `maxSockets`: el mismo valor que estableció para `maxConcurrentTransactions`

  Para ver un ejemplo, consulte el siguiente código de JavaScript o TypeScript.

  JavaScript

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

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

const agentForQldb = new https.Agent({
    "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);
```

- Otra opción consiste en establecer la variable del entorno `AWS_NODEJS_CONNECTION_REUSE_ENABLED` en 1. Para obtener más información, consulte Reutilización de conexiones con KeepAlive en Node.js en la Guía para desarrolladores de AWS SDK for JavaScript.

> **Note**
>
> Si define esta variable de entorno, afectará a todos los Servicios de AWS que usen AWS SDK for JavaScript.

Controlador Amazon QLDB para Node.js: tutorial de inicio rápido

En este tutorial aprenderá a configurar una aplicación sencilla con la última versión del controlador Amazon QLDB para Node.js. En esta guía se incluyen los pasos para instalar el controlador y ejemplos de código JavaScript y TypeScript breves de las operaciones básicas de creación, lectura, actualización y eliminación (CRUD). Para ver ejemplos más detallados que presentan estas operaciones en una aplicación de muestra completa, consulte Tutorial de Node.js.

> **Note**
>
> Cuando proceda, algunos pasos incluyen ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Node.js.

Temas

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Si aún no lo ha hecho, complete el Requisitos previos para el controlador Node.js. Deberá registrarse en AWS, conceder acceso programático de desarrollo e instalar Node.js.
2. Cree un libro mayor denominado `quick-start`.

   Para obtener más información sobre cómo crear un libro mayor, consulte Operaciones básicas de libros mayores de Amazon QLDB o Paso 1: crear un nuevo libro mayor en introducción a la consola.

Si utiliza TypeScript, también debe realizar los siguientes pasos de configuración.

Utilizar TypeScript

Para instalar TypeScript

1. Instale el paquete TypeScript. El controlador QLDB se ejecuta en TypeScript 3.8.x.

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

2. Una vez instalado el paquete, ejecute el siguiente comando para asegurarse de que el compilador de TypeScript está instalado.

   ```
   $ tsc --version
   ```

Para ejecutar el código en los siguientes pasos, tenga en cuenta que primero debe transpilar el archivo TypeScript en código JavaScript ejecutable, de la siguiente manera.

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

Paso 1: Configuración del proyecto

En primer lugar, configure su proyecto de Node.js.

1. Cree una carpeta para su aplicación.

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

2. Para inicializar el proyecto, introduzca el siguiente comando `npm` y responda a las preguntas que se le formulen durante la configuración. Puede utilizar los valores predeterminados para la mayoría de las preguntas.

```
$ npm init
```

3. Instale el controlador Amazon QLDB para Node.js.
   - Uso de la versión 3.x

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

   - Uso de la versión 2.x

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

   - Uso de la versión 1.x

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

4. Instale las dependencias homólogas del controlador.
   - Uso de la versión 3.x

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

   - Utilizar la versión 2.x o 1.x

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

5. Cree un nuevo archivo con el nombre `app.js` para JavaScript o `app.ts` para TypeScript.

   A continuación, añada gradualmente los ejemplos de código en los siguientes pasos para probar algunas operaciones básicas de CRUD. También puede saltarse el tutorial paso a paso y ejecutar la aplicación completa.

Paso 2: inicializar el controlador

Inicialice una instancia del controlador que se conecte al libro mayor denominado `quick-start`. Agregue el siguiente código a su archivo `app.js` o `app.ts`.

Uso de la versión 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

- En este ejemplo de código, sustituya *us-east-1* por el Región de AWS donde creó el libro mayor.
- Para simplificar, el resto de los ejemplos de código de esta guía utilizan un controlador con la configuración predeterminada, tal y como se especifica en el siguiente ejemplo de la versión 1.x. También puede utilizar su propia instancia de controlador con una `RetryConfig` personalizada en su lugar.

Uso de la versión 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

- En este ejemplo de código, sustituya *us-east-1* por el Región de AWS donde creó el libro mayor.
- La versión 2.x introduce el nuevo parámetro opcional `RetryConfig` para la inicialización de `QldbDriver`.
- Para simplificar, el resto de los ejemplos de código de esta guía utilizan un controlador con la configuración predeterminada, tal y como se especifica en el siguiente ejemplo de la versión 1.x. También puede utilizar su propia instancia de controlador con una `RetryConfig` personalizada en su lugar.
- Este ejemplo de código inicializa un controlador que reutiliza las conexiones existentes mediante la configuración de opciones keep-alive. Para obtener más información, consulte Recomendaciones de configuración para el controlador de Node.js.

Uso de la versión 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**
>
> Puede establecer la región mediante la variable de entorno `AWS_REGION`. Para obtener más información, consulte Configurar Región de AWS en la Guía para desarrolladores de AWS SDK for JavaScript.

Paso 3: crear una tabla y un índice

Los siguientes ejemplos de código muestran cómo ejecutar las instrucciones `CREATE TABLE` y `CREATE INDEX`.

1. Agregue la siguiente función que crea una tabla llamada `People`.

JavaScript

```
async function createTable(txn) {
    await txn.execute("CREATE TABLE People");
}
```

TypeScript

```
async function createTable(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE TABLE People");
}
```

2. Agregue la siguiente función que crea un índice para el campo `firstName` de la tabla `People`. Los índices son necesarios para optimizar el rendimiento de las consultas y ayudar a limitar las excepciones de conflicto de control de concurrencia optimista (OCC).

   JavaScript

   ```
   async function createIndex(txn) {
       await txn.execute("CREATE INDEX ON People (firstName)");
   }
   ```

   TypeScript

   ```
   async function createIndex(txn: TransactionExecutor): Promise<void> {
       await txn.execute("CREATE INDEX ON People (firstName)");
   }
   ```

3. En la función `main`, primero llame a `createTable` y después llame a `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. Ejecute el código para crear la tabla y el índice.

JavaScript

```
$ node app.js
```

TypeScript

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

Paso 4: insertar un documento

Los siguientes ejemplos de código muestran cómo ejecutar a instrucción `INSERT`. QLDB es compatible con el lenguaje de consultas PartiQL (compatible con SQL) y el formato de datos Amazon Ion (superconjunto de JSON).

1. Añada la siguiente función para insertar un documento en la tabla `People`.

   JavaScript

   ```
   async function insertDocument(txn) {
       const person = {
           firstName: "John",
           lastName: "Doe",
           age: 42
       };
       await txn.execute("INSERT INTO People ?", person);
   }
   ```

   TypeScript

   ```
   async function insertDocument(txn: TransactionExecutor): Promise<void> {
       const person: Record<string, any> = {
           firstName: "John",
           lastName: "Doe",
           age: 42
       };
       await txn.execute("INSERT INTO People ?", person);
   }
   ```

   En este ejemplo se emplea un signo de interrogación (?) como marcador de posición variable para pasar la información del documento a la instrucción. El método `execute` admite valores tanto en los tipos de Amazon Ion como en los tipos nativos de Node.js.

   > **Tip**
   >
   > Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo list a la instrucción de la siguiente manera.
   >
   > ```
   > // people is a list
   > ```

```
txn.execute("INSERT INTO People ?", people);
```

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (<<...>>) al pasar una lista. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

2. En la función `main`, elimine las llamadas `createTable` y `createIndex` y añada una llamada a `insertDocument`.

JavaScript

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

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    await driver.executeLambda(async (txn) => {
        console.log("Insert document");
        await insertDocument(txn);
    });

    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    await driver.executeLambda(async (txn: TransactionExecutor) => {
        console.log("Insert document");
        await insertDocument(txn);
    });

    driver.close();
}
```

```
if (require.main === module) {
    main();
}
```

Paso 5: consultar al documento

El siguiente ejemplo de código muestra cómo ejecutar una instrucción `SELECT`.

1. Agregue la siguiente función para consultar un documento de la tabla `People`.

   JavaScript

   ```
   async function fetchDocuments(txn) {
       return await txn.execute("SELECT firstName, age, lastName FROM People WHERE
    firstName = ?", "John");
   }
   ```

   TypeScript

   ```
   async function fetchDocuments(txn: TransactionExecutor): Promise<dom.Value[]> {
       return (await txn.execute("SELECT firstName, age, lastName FROM People WHERE
    firstName = ?", "John")).getResultList();
   }
   ```

2. En la función `main`, añada la siguiente llamada a `fetchDocuments` después de la llamada a `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();
}
```

Paso 6: actualizar el documento

Los siguientes ejemplos de código muestran cómo ejecutar a instrucción `UPDATE`.

1. Añada la siguiente función para actualizar un documento de la tabla `People` actualizando `lastName` a `"Stiles"`.

JavaScript

```
async function updateDocuments(txn) {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}
```

TypeScript

```
async function updateDocuments(txn: TransactionExecutor): Promise<void> {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}
```

2. En la función `main`, añada la siguiente llamada a `updateDocuments` después de la llamada a `fetchDocuments`. A continuación, vuelva a llamar a `fetchDocuments` para ver los resultados actualizados.

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. Ejecute el código para insertar, consultar y actualizar un documento.

JavaScript

```
$ node app.js
```

TypeScript

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

Ejecución de la aplicación completa

Los siguientes ejemplos de código son las versiones completas de `app.js` y `app.ts`. En lugar de seguir los pasos anteriores de forma individual, también puede ejecutar este ejemplo de código de principio a fin. Esta aplicación muestra algunas operaciones básicas de CRUD en el libro mayor denominado `quick-start`.

> **Note**
>
> Antes de ejecutar este código, asegúrese de no tener ya una tabla activa con el nombre `People` en el libro mayor `quick-start`.

JavaScript

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

async function createTable(txn) {
    await txn.execute("CREATE TABLE People");
}

async function createIndex(txn) {
    await txn.execute("CREATE INDEX ON People (firstName)");
}

async function insertDocument(txn) {
    const person = {
        firstName: "John",
        lastName: "Doe",
        age: 42
    };
    await txn.execute("INSERT INTO People ?", person);
}

async function fetchDocuments(txn) {
```

```
    return await txn.execute("SELECT firstName, age, lastName FROM People WHERE
 firstName = ?", "John");
}

async function updateDocuments(txn) {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
}

async function main() {
    // Use default settings
    const driver = new qldb.QldbDriver("quick-start");

    var resultList = await driver.executeLambda(async (txn) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        var result = await fetchDocuments(txn);
        return result.getResultList();
    });

    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

main();
```

TypeScript

```
import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";
import { dom } from "ion-js";

async function createTable(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE TABLE People");
```

```
}

async function createIndex(txn: TransactionExecutor): Promise<void> {
    await txn.execute("CREATE INDEX ON People (firstName)");
}

async function insertDocument(txn: TransactionExecutor): Promise<void> {
    const person: Record<string, any> = {
        firstName: "John",
        lastName: "Doe",
        age: 42
    };
    await txn.execute("INSERT INTO People ?", person);
}

async function fetchDocuments(txn: TransactionExecutor): Promise<dom.Value[]> {
    return (await txn.execute("SELECT firstName, age, lastName FROM People WHERE
 firstName = ?", "John")).getResultList();
}

async function updateDocuments(txn: TransactionExecutor): Promise<void> {
    await txn.execute("UPDATE People SET lastName = ? WHERE firstName = ?",
 "Stiles", "John");
};

async function main(): Promise<void> {
    // Use default settings
    const driver: QldbDriver = new QldbDriver("quick-start");

    const resultList: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor) => {
        console.log("Create table People");
        await createTable(txn);
        console.log("Create index on firstName");
        await createIndex(txn);
        console.log("Insert document");
        await insertDocument(txn);
        console.log("Fetch document");
        await fetchDocuments(txn);
        console.log("Update document");
        await updateDocuments(txn);
        console.log("Fetch document after update");
        return await fetchDocuments(txn);
    });
```

```
    // Pretty print the result list
    console.log("The result List is ", JSON.stringify(resultList, null, 2));
    driver.close();
}

if (require.main === module) {
    main();
}
```

Introduzca el siguiente comando para ejecutar la aplicación completa.

JavaScript

```
$ node app.js
```

TypeScript

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

Controlador Amazon QLDB para Node.js: libro de recetas de referencia

Esta guía de referencia muestra los casos de uso más comunes del controlador Amazon QLDB para Node.js. En él se proporcionan ejemplos de código JavaScript y TypeScript que muestran cómo utilizar el controlador para ejecutar operaciones básicas de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés). También incluye ejemplos de código para procesar datos de Amazon Ion. Además, esta guía destaca las mejores prácticas para hacer que las transacciones sean idempotentes e implementar restricciones de exclusividad.

Contenido

Importación del controlador

En los ejemplos de código se utilizan las siguientes importaciones.

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**
>
> En este ejemplo también se importa el paquete Amazon Ion (`ion-js`). Necesita este paquete para procesar los datos de Ion al ejecutar algunas operaciones de datos de esta referencia. Para obtener más información, consulte Trabajar con Amazon Ion.

Instanciación del controlador

En el siguiente ejemplo de código, se crea una instancia del controlador que se conecta a un nombre de libro mayor especificado mediante la configuración predeterminada.

JavaScript

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

TypeScript

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

Operaciones CRUD

La QLDB ejecuta operaciones de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés) como parte de una transacción.

> **Warning**
>
> Como práctica recomendada, haga que sus transacciones de escritura sean estrictamente idempotentes.

Hacer que las transacciones sean idempotentes

Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos. Una transacción es idempotente si puede ejecutarse varias veces y producir resultados idénticos cada vez.

Por ejemplo, pensemos en una transacción que inserta un documento en una tabla llamada `Person`. La transacción debe comprobar primero si el documento ya existe o no en la tabla. Sin esta comprobación, la tabla podría terminar con documentos duplicados.

Supongamos que la QLDB confirma correctamente la transacción en el lado del servidor pero el cliente agota el tiempo de espera mientras espera una respuesta. Si la transacción no es idempotente, se podría insertar el mismo documento más de una vez en caso de volver a intentarlo.

Uso de índices para evitar escanear tablas completas

También le recomendamos que ejecute instrucciones con una frase de predicado `WHERE` utilizando un operador de igualdad sobre un campo indexado o un ID de documento; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Sin esta búsqueda indexada, la QLDB necesita escanear las tablas, lo que puede provocar tiempos de espera de las transacciones o conflictos de control de concurrencia optimista (OCC).

Para obtener más información acerca de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Transacciones creadas de forma implícita

La función QldbDriver.executeLambda acepta una función de Lambda que recibe una instancia de TransactionExecutor, que se puede utilizar para ejecutar instrucciones. La instancia de `TransactionExecutor` envuelve una transacción creada de forma implícita.

Puede ejecutar instrucciones dentro de la función de Lambda mediante el método execute de transacciones. El controlador confirma implícitamente la transacción cuando vuelve la función de Lambda.

> **Note**
>
> El método `execute` admite valores tanto en los tipos de Amazon Ion como en los tipos nativos de Node.js. Si pasa un tipo nativo de Node.js como argumento a `execute`, el controlador lo convierte en un tipo Ion mediante el paquete `ion-js` (siempre que se admita la conversión para el tipo de datos Node.js indicado). Para conocer los tipos de datos y las reglas de conversión compatibles, consulte el README del DOM de Ion JavaScript.

En las siguientes secciones se muestra cómo ejecutar operaciones CRUD básicas, especificar una lógica de reintento personalizada e implementar restricciones de exclusividad.

Contenido

- Eliminación de documentos
- Ejecutar varias instrucciones en una transacción
- Lógica de reintentos
- Implementación de restricciones de exclusividad

Creación de tablas

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

Creación de índices

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute("CREATE INDEX ON Person (GovId)");
    });
})();
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('CREATE INDEX ON Person (GovId)');
    });
```

```
}());
```

Lectura de documentos

JavaScript

```
(async function() {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'")).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute("SELECT * FROM Person WHERE
 GovId = 'TOYENC486FH'")).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

Uso de parámetros de consulta

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo nativo.

JavaScript

```
(async function() {
```

```
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    // Assumes that Person table has documents as follows:
    // { "GovId": "TOYENC486FH", "FirstName": "Brent" }
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govId = ionjs.load("TOYENC486FH");

        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 govId)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
```

```
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govId: dom.Value = load("TOYENC486FH");

        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', govId)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

En el siguiente ejemplo de código se utilizan varios parámetros de consulta.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ? AND
 FirstName = ?', 'TOYENC486FH', 'Brent')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ? AND FirstName = ?', 'TOYENC486FH', 'Brent')).getResultList();
        for (let result of results) {
            console.log(result.get('GovId')); // prints [String: 'TOYENC486FH']
            console.log(result.get('FirstName')); // prints [String: 'Brent']
        }
```

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

En el siguiente ejemplo de código, se utiliza una lista de parámetros de consulta.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govIds = ['TOYENC486FH','LOGANB486CG','LEWISR261LL'];
        /*
        Assumes that Person table has documents as follows:
        { "GovId": "TOYENC486FH", "FirstName": "Brent" }
        { "GovId": "LOGANB486CG", "FirstName": "Brent" }
        { "GovId": "LEWISR261LL", "FirstName": "Raul" }
        */
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId IN
 (?,?,?)', ...govIds)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId'));
            console.log(result.get('FirstName'));
            /*
            prints:
            [String: 'TOYENC486FH']
            [String: 'Brent']
            [String: 'LOGANB486CG']
            [String: 'Brent']
            [String: 'LEWISR261LL']
            [String: 'Raul']
            */
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govIds: string[] = ['TOYENC486FH','LOGANB486CG','LEWISR261LL'];
        /*
        Assumes that Person table has documents as follows:
        { "GovId": "TOYENC486FH", "FirstName": "Brent" }
```

```
        { "GovId": "LOGANB486CG", "FirstName": "Brent" }
        { "GovId": "LEWISR261LL", "FirstName": "Raul" }
        */
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId IN (?,?,?)', ...govIds)).getResultList();
        for (let result of results) {
            console.log(result.get('GovId'));
            console.log(result.get('FirstName'));
            /*
            prints:
            [String: 'TOYENC486FH']
            [String: 'Brent']
            [String: 'LOGANB486CG']
            [String: 'Brent']
            [String: 'LEWISR261LL']
            [String: 'Raul']
            */
        }
    });
}());
```

> **Note**
>
> Cuando ejecuta una consulta sin una búsqueda indexada, se invoca un escaneo completo de la tabla. En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las consultas pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Inserción de documentos

El siguiente ejemplo de código inserta los tipos de datos nativos.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
```

```
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            await txn.execute('INSERT INTO Person ?', doc);
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc: Record<string, string> = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            await txn.execute('INSERT INTO Person ?', doc);
        }
    });
}());
```

El siguiente ejemplo de código inserta los tipos de datos de Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results = (await txn.execute('SELECT * FROM Person WHERE GovId = ?',
 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
```

```
            const doc = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            // Create a sample Ion doc
            const ionDoc = ionjs.load(ionjs.dumpBinary(doc));

            await txn.execute('INSERT INTO Person ?', ionDoc);
        }
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        // Check if doc with GovId:TOYENC486FH exists
        // This is critical to make this transaction idempotent
        const results: dom.Value[] = (await txn.execute('SELECT * FROM Person WHERE
 GovId = ?', 'TOYENC486FH')).getResultList();
        // Insert the document after ensuring it doesn't already exist
        if (results.length == 0) {
            const doc: Record<string, string> = {
                'FirstName': 'Brent',
                'GovId': 'TOYENC486FH',
            };
            // Create a sample Ion doc
            const ionDoc: dom.Value = load(dumpBinary(doc));

            await txn.execute('INSERT INTO Person ?', ionDoc);
        }
    });
}());
```

Esta transacción inserta un documento en la tabla `Person`. Antes de insertar, compruebe primero si el documento ya existe en la tabla. Esta comprobación hace que la transacción sea de naturaleza idempotente. Incluso si realiza esta transacción varias veces, no provocará ningún efecto secundario no deseado.

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Insertar varios documentos en una instrucción

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo list a la instrucción de la siguiente manera.

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

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (`<<...>>`) al pasar una lista. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Actualización de documentos

El siguiente ejemplo de código utiliza tipos de datos nativos.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?', 'John',
 'TOYENC486FH');
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?', 'John',
 'TOYENC486FH');
    });
}());
```

El siguiente ejemplo de código utiliza tipos de datos Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const firstName = ionjs.load("John");
        const govId = ionjs.load("TOYENC486FH");

        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?',
 firstName, govId);
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const firstName: dom.Value = load("John");
        const govId: dom.Value = load("TOYENC486FH");

        await txn.execute('UPDATE Person SET FirstName = ? WHERE GovId = ?',
 firstName, govId);
    });
}());
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Eliminación de documentos

El siguiente ejemplo de código utiliza tipos de datos nativos.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
```

```
        await txn.execute('DELETE FROM Person WHERE GovId = ?', 'TOYENC486FH');
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        await txn.execute('DELETE FROM Person WHERE GovId = ?', 'TOYENC486FH');
    });
}());
```

El siguiente ejemplo de código utiliza tipos de datos Ion.

JavaScript

```
(async function() {
    await qldbDriver.executeLambda(async (txn) => {
        const govId = ionjs.load("TOYENC486FH");

        await txn.execute('DELETE FROM Person WHERE GovId = ?', govId);
    });
}());
```

TypeScript

```
(async function(): Promise<void> {
    await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
        const govId: dom.Value = load("TOYENC486FH");

        await txn.execute('DELETE FROM Person WHERE GovId = ?', govId);
    });
}());
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Ejecutar varias instrucciones en una transacción

TypeScript

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
async function insureCar(driver: QldbDriver, vin: string): Promise<boolean> {

    return await driver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 vin)).getResultList();

        if (results.length > 0) {
            await txn.execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin);
            return true;
        }
        return false;
    });
};
```

Lógica de reintentos

El método del controlador `executeLambda` tiene un mecanismo de reintento integrado que reintenta la transacción si se produce una excepción que se pueda volver a intentar (como tiempos de espera o conflictos de OCC). El número máximo de reintentos y la estrategia de retardo se pueden configurar.

El límite de reintentos predeterminado es `4` y la estrategia de retardo predeterminada es defaultBackoffFunction, con una base de `10` milisegundos. Puede establecer la configuración de reintentos por instancia de controlador y también por transacción mediante una instancia de RetryConfig.

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una instancia del controlador.

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);
```

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una ejecución lambda particular. Esta configuración `executeLambda` anula la lógica de reintentos establecida para la instancia del controlador.

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);
}());
```

Implementación de restricciones de exclusividad

QLDB no admite índices únicos, pero puede implementar este comportamiento en su aplicación.

Suponga que desea implementar una restricción de exclusividad en el campo `GovId` de la tabla `Person`. Para ello, puede escribir una transacción que haga lo siguiente:

1. Afirmar que en la tabla no hay documentos existentes con un `GovId` especificado.
2. Insertar el documento si se aprueba la afirmación.

Si una transacción competidora supera la afirmación simultáneamente, solo una de las transacciones se confirmará correctamente. La otra transacción fallará y se producirá una excepción de conflicto de OCC.

En el siguiente ejemplo de código, se muestra cómo implementar esta lógica de restricción de exclusividad.

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**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Trabajar con Amazon Ion

Las siguientes secciones muestran cómo usar el módulo Amazon Ion para procesar los datos de Ion.

Contenido

Importación del módulo Ion

JavaScript

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

TypeScript

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

Creación de tipos de Ion

El siguiente ejemplo de código crea un objeto Ion a partir del texto de Ion.

JavaScript

```
const ionText  = '{GovId: "TOYENC486FH", FirstName: "Brent"}';
const ionObj = ionjs.load(ionText);

console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

TypeScript

```
const ionText: string = '{GovId: "TOYENC486FH", FirstName: "Brent"}';
const ionObj: dom.Value = load(ionText);

console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

El siguiente ejemplo de código crea un objeto Ion a partir del diccionario de Node.js.

JavaScript

```
const aDict = {
    'GovId': 'TOYENC486FH',
    'FirstName': 'Brent'
};
const ionObj = ionjs.load(ionjs.dumpBinary(aDict));
```

```
console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

TypeScript

```
const aDict: Record<string, string> = {
    'GovId': 'TOYENC486FH',
    'FirstName': 'Brent'
};
const ionObj: dom.Value = load(dumpBinary(aDict));
console.log(ionObj.get('GovId')); // prints [String: 'TOYENC486FH']
console.log(ionObj.get('FirstName')); // prints [String: 'Brent']
```

Obtener un volcado binario de Ion

JavaScript

```
// ionObj is an Ion struct
console.log(ionjs.dumpBinary(ionObj).toString()); // prints
 224,1,0,234,238,151,129,131,222,147,135,190,144,133,71,111,118,73,100,137,70,105,114,115,11
```

TypeScript

```
// ionObj is an Ion struct
console.log(dumpBinary(ionObj).toString()); // prints
 224,1,0,234,238,151,129,131,222,147,135,190,144,133,71,111,118,73,100,137,70,105,114,115,11
```

Obtener un volcado de texto de Ion

JavaScript

```
// ionObj is an Ion struct
console.log(ionjs.dumpText(ionObj)); // prints
 {GovId:"TOYENC486FH",FirstName:"Brent"}
```

TypeScript

```
// ionObj is an Ion struct
console.log(dumpText(ionObj)); // prints {GovId:"TOYENC486FH",FirstName:"Brent"}
```

Para obtener más información acerca de Ion, consulte la documentación de Amazon Ion en GitHub. Para ver más ejemplos de código sobre cómo trabajar con Ion en QLDB, consulte Uso de tipos de datos de Amazon Ion en Amazon QLDB.

Controlador Amazon QLDB para Python

Para trabajar con los datos de su libro mayor, puede conectarse a Amazon QLDB desde su aplicación Python mediante un controlador proporcionado por AWS. En los siguientes temas se describe cómo empezar a usar el controlador QLDB para Python.

Temas

Recursos de controladores

Para obtener más información sobre las funcionalidades compatibles con el controlador Python, consulte los siguientes recursos:

- Referencia de API: 3.x, 2.x
- Código origen del controlador (GitHub)
- Ejemplo de código origen de aplicación (GitHub)
- Ejemplos de código de Amazon Ion

Requisitos previos

Antes de empezar a usar el controlador QLDB para Python, debe hacer lo siguiente:

1. Siga las instrucciones de configuración de AWS en Acceso a Amazon QLDB. Estas incluyen las siguientes:

    1. Regístrese en AWS.
    2. Cree un usuario con los permisos de QLDB adecuados.

3. Conceda acceso programático de desarrollo.

2. Descargue una de las siguientes versiones de Python desde el sitio de descargas de Python:

   - 3.6 o posterior: controlador de QLDB para Python v3
   - 3.4 o posterior: controlador de QLDB para Python v2

3. Configure sus credenciales AWS y sus Región de AWS predeterminadas. Para obtener instrucciones, consulte inicio rápido en la documentación AWS SDK for Python (Boto3).

   Para ver una lista completa de las regiones disponibles, consulte puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

A continuación, puede descargar la aplicación de ejemplo completa del tutorial, o bien instalar solo el controlador en un proyecto de Python y ejecutar ejemplos de códigos cortos.

- Para instalar el controlador QLDB y AWS SDK for Python (Boto3) en un proyecto existente, acceda a Instalación.
- Para configurar un proyecto y ejecutar ejemplos de códigos cortos que muestren las transacciones de datos básicas en un libro mayor, consulte Tutorial de inicio rápido.
- Para ver ejemplos más detallados de las operaciones de la API de datos y administración en la aplicación de ejemplo completa del tutorial, consulte Tutorial de Python.

Instalación

QLDB es compatible con las siguientes versiones de controlador y sus dependencias de Python.

Versión de controlador	Versión de Python	Estado	Fecha de lanzamiento más reciente
2.x	3.4 o posterior	Lanzamiento de producción	7 de mayo de 2020
3.x	3.6 o posterior	Lanzamiento de producción	28 de octubre de 2021

Para instalar el controlador QLDB desde PyPI con `pip` (un administrador de paquetes para Python), introduzca la siguiente línea de comando.

3.x

```
pip install pyqldb
```

2.x

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

Al instalar el controlador también se instalan sus dependencias, incluidos AWS SDK for Python (Boto3) y los paquetes de Amazon Ion.

Uso del controlador para conectar a un libro mayor

A continuación, puede importar el controlador y usarlo para conectar a un libro mayor. En el siguiente ejemplo de código de Python se muestra cómo crear una sesión para un nombre de libro mayor específico.

3.x

```
from pyqldb.driver.qldb_driver import QldbDriver
qldb_driver = QldbDriver(ledger_name='testLedger')

for table in qldb_driver.list_tables():
    print(table)
```

2.x

```
from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver

qldb_driver = PooledQldbDriver(ledger_name='testLedger')
qldb_session = qldb_driver.get_session()

for table in qldb_session.list_tables():
    print(table)
```

Para ver ejemplos de códigos cortos sobre cómo ejecutar transacciones de datos básicos en un libro mayor, consulte la Referencia de libro de recetas.

Controlador Amazon QLDB para Python: tutorial de inicio rápido

En este tutorial aprenderá a configurar una aplicación sencilla con la última versión del controlador Amazon QLDB para Python. En esta guía se incluyen los pasos para instalar el controlador y ejemplos de código breve de las operaciones básicas de creación, lectura, actualización y eliminación (CRUD). Para ver ejemplos más detallados que presentan estas operaciones en una aplicación de muestra completa, consulte Tutorial de Python.

Temas

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Si aún no lo ha hecho, complete el Requisitos previos para el controlador Python. Deberá registrarse en AWS, conceder acceso programático de desarrollo e instalar Python versión 3.6 o posterior.
2. Cree un libro mayor denominado `quick-start`.

   Para obtener más información sobre cómo crear un libro mayor, consulte Operaciones básicas de libros mayores de Amazon QLDB o Paso 1: crear un nuevo libro mayor en Introducción a la consola.

Paso 1: Configuración del proyecto

En primer lugar, configure su proyecto de Python.

> **Note**
>
> Si usa un IDE con características para automatizar estos pasos de configuración, puede pasar directamente a Paso 2: inicializar el controlador.

1. Cree una carpeta para su aplicación.

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

2. Para instalar el controlador QLDB para Python desde PyPI, introduzca el comando `pip`.

```
$ pip install pyqldb
```

   Al instalar el controlador también se instalan sus dependencias, incluidos AWS SDK for Python (Boto3) y los paquetes de Amazon Ion.

3. Cree un nuevo archivo llamado `app.py`.

   A continuación, añada gradualmente los ejemplos de código en los siguientes pasos para probar algunas operaciones básicas de CRUD. También puede saltarse el tutorial paso a paso y ejecutar la aplicación completa.

Paso 2: inicializar el controlador

Inicialice una instancia del controlador que se conecte al libro mayor denominado `quick-start`. Agregue el siguiente código al archivo `app.py`.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

# Configure retry limit to 3
retry_config = RetryConfig(retry_limit=3)

# Initialize the driver
print("Initializing the driver")
qldb_driver = QldbDriver("quick-start", retry_config=retry_config)
```

Paso 3: crear una tabla y un índice

Los siguientes ejemplos de código muestran cómo ejecutar las instrucciones `CREATE TABLE` y `CREATE INDEX`.

Agregue el siguiente código para crear una tabla con el nombre `People` y un índice para el campo `lastName` de dicha tabla. Los índices son necesarios para optimizar el rendimiento de las consultas y ayudar a limitar las excepciones de conflicto de control de concurrencia optimista (OCC).

```
def create_table(transaction_executor):
    print("Creating a table")
    transaction_executor.execute_statement("Create TABLE People")

def create_index(transaction_executor):
    print("Creating an index")
    transaction_executor.execute_statement("CREATE INDEX ON People(lastName)")

# Create a table
qldb_driver.execute_lambda(lambda executor: create_table(executor))

# Create an index on the table
qldb_driver.execute_lambda(lambda executor: create_index(executor))
```

Paso 4: insertar un documento

Los siguientes ejemplos de código muestran cómo ejecutar una instrucción de `INSERT`. QLDB es compatible con el lenguaje de consultas PartiQL (compatible con SQL) y el formato de datos Amazon Ion (superconjunto de JSON).

Añada el siguiente código para insertar un documento en la tabla `People`.

```
def insert_documents(transaction_executor, arg_1):
    print("Inserting a document")
    transaction_executor.execute_statement("INSERT INTO People ?", arg_1)

# Insert a document
doc_1 = { 'firstName': "John",
          'lastName': "Doe",
          'age': 32,
        }
```

```
qldb_driver.execute_lambda(lambda x: insert_documents(x, doc_1))
```

En este ejemplo se emplea un signo de interrogación (?) como marcador de posición variable para pasar la información del documento a la instrucción. El método `execute_statement` admite valores tanto en los tipos de Amazon Ion como en los tipos nativos de Python.

> **Tip**
>
> Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo list a la instrucción de la siguiente manera.
>
> ```
> # people is a list
> transaction_executor.execute_statement("INSERT INTO Person ?", people)
> ```
>
> No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (`<<...>>`) al pasar una lista. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Paso 5: consulta del documento

El siguiente ejemplo de código muestra cómo ejecutar una instrucción de `SELECT`.

Agregue el siguiente código para consultar un documento de la tabla `People`.

```
def read_documents(transaction_executor):
    print("Querying the table")
    cursor = transaction_executor.execute_statement("SELECT * FROM People WHERE
 lastName = ?", 'Doe')

    for doc in cursor:
        print(doc["firstName"])
        print(doc["lastName"])
        print(doc["age"])

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

Paso 6: actualizar el documento

Los siguientes ejemplos de código muestran cómo ejecutar una instrucción de `UPDATE`.

1. Añada el siguiente código para actualizar un documento de la tabla `People` actualizando `age` a `42`.

```
def update_documents(transaction_executor, age, lastName):
    print("Updating the document")
    transaction_executor.execute_statement("UPDATE People SET age = ? WHERE
 lastName = ?", age, lastName)

# Update the document
age = 42
lastName = 'Doe'

qldb_driver.execute_lambda(lambda x: update_documents(x, age, lastName))
```

2. Vuelva a consultar la tabla para ver el valor actualizado.

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

3. Para ejecutar la aplicación, introduzca el siguiente comando desde el directorio del proyecto.

```
$ python app.py
```

Ejecución de la aplicación completa

El siguiente ejemplo de código es la versión completa de la aplicación `app.py`. En lugar de seguir los pasos anteriores de forma individual, también puede copiar y ejecutar este ejemplo de código de principio a fin. Esta aplicación muestra algunas operaciones básicas de CRUD en el libro mayor denominado `quick-start`.

> **Note**
>
> Antes de ejecutar este código, asegúrese de no tener ya una tabla activa con el nombre `People` en el libro mayor `quick-start`.

```
from pyqldb.config.retry_config import RetryConfig
from pyqldb.driver.qldb_driver import QldbDriver

def create_table(transaction_executor):
```

```
    print("Creating a table")
    transaction_executor.execute_statement("CREATE TABLE People")

def create_index(transaction_executor):
    print("Creating an index")
    transaction_executor.execute_statement("CREATE INDEX ON People(lastName)")

def insert_documents(transaction_executor, arg_1):
    print("Inserting a document")
    transaction_executor.execute_statement("INSERT INTO People ?", arg_1)

def read_documents(transaction_executor):
    print("Querying the table")
    cursor = transaction_executor.execute_statement("SELECT * FROM People WHERE
 lastName = ?", 'Doe')


    for doc in cursor:
        print(doc["firstName"])
        print(doc["lastName"])
        print(doc["age"])

def update_documents(transaction_executor, age, lastName):
    print("Updating the document")
    transaction_executor.execute_statement("UPDATE People SET age = ? WHERE lastName
 = ?", age, lastName)

# Configure retry limit to 3
retry_config = RetryConfig(retry_limit=3)

# Initialize the driver
print("Initializing the driver")
qldb_driver = QldbDriver("quick-start", retry_config=retry_config)

# Create a table
qldb_driver.execute_lambda(lambda executor: create_table(executor))

# Create an index on the table
qldb_driver.execute_lambda(lambda executor: create_index(executor))

# Insert a document
doc_1 = { 'firstName': "John",
          'lastName': "Doe",
          'age': 32,
```

```
        }

qldb_driver.execute_lambda(lambda x: insert_documents(x, doc_1))

# Query the table
qldb_driver.execute_lambda(lambda executor: read_documents(executor))

# Update the document
age = 42
lastName = 'Doe'

qldb_driver.execute_lambda(lambda x: update_documents(x, age, lastName))

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

Para ejecutar la aplicación completa, introduzca el siguiente comando desde el directorio de su proyecto.

```
$ python app.py
```

Controlador Amazon QLDB para Python: libro de recetas de referencia

Esta guía de referencia muestra los casos de uso más comunes del controlador Amazon QLDB para Python. En él se proporcionan ejemplos de código Python que muestran cómo utilizar el controlador para ejecutar operaciones básicas de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés). También incluye ejemplos de código para procesar datos de Amazon Ion. Además, esta guía destaca las mejores prácticas para hacer que las transacciones sean idempotentes e implementar restricciones de exclusividad.

> **Note**
>
> Cuando proceda, algunos pasos incluyen ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Python.

Contenido

Importación del controlador

En los ejemplos de código se utilizan las siguientes importaciones.

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**
>
> En este ejemplo también se importa el paquete Amazon Ion (`amazon.ion.simpleion`). Necesita este paquete para procesar los datos de Ion al ejecutar algunas operaciones de datos de esta referencia. Para obtener más información, consulte Trabajar con Amazon Ion.

Instanciación del controlador

En el siguiente ejemplo de código, se crea una instancia del controlador que se conecta a un nombre de libro mayor especificado mediante la configuración predefinida.

3.x

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

2.x

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

Operaciones CRUD

La QLDB ejecuta operaciones de creación, lectura, actualización y eliminación (CRUD, por sus siglas en inglés) como parte de una transacción.

> **Warning**
>
> Como práctica recomendada, haga que sus transacciones de escritura sean estrictamente idempotentes.

Hacer que las transacciones sean idempotentes

Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos. Una transacción es idempotente si puede ejecutarse varias veces y producir resultados idénticos cada vez.

Por ejemplo, pensemos en una transacción que inserta un documento en una tabla llamada `Person`. La transacción debe comprobar primero si el documento ya existe o no en la tabla. Sin esta comprobación, la tabla podría terminar con documentos duplicados.

Supongamos que la QLDB confirma correctamente la transacción en el lado del servidor pero el cliente agota el tiempo de espera mientras espera una respuesta. Si la transacción no es idempotente, se podría insertar el mismo documento más de una vez en caso de volver a intentarlo.

Uso de índices para evitar escanear tablas completas

También le recomendamos que ejecute instrucciones con una frase de predicado `WHERE` utilizando un operador de igualdad sobre un campo indexado o un ID de documento; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Sin esta búsqueda indexada, la QLDB necesita escanear las tablas, lo que puede provocar tiempos de espera de las transacciones o conflictos de control de concurrencia optimista (OCC).

Para obtener más información acerca de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Transacciones creadas de forma implícita

El método pyqldb.driver.qldb_driver.execute_lambda acepta una función de Lambda que recibe una instancia de pyqldb.execution.executor.Executor, que se puede utilizar para ejecutar instrucciones. La instancia de `Executor` envuelve una transacción creada de forma implícita.

Puede ejecutar instrucciones dentro de la función de Lambda mediante el método execute_statement de transacciones. El controlador confirma implícitamente la transacción cuando vuelve la función de Lambda.

> **Note**
>
> El método `execute_statement` admite valores tanto en los tipos de Amazon Ion como en los tipos nativos de Python. Si pasa un tipo nativo de Python como argumento a `execute_statement`, el controlador lo convierte en un tipo Ion mediante el paquete `amazon.ion.simpleion`(siempre que se admita la conversión para el tipo de datos Python indicado). Para conocer los tipos de datos y las reglas de conversión compatibles, consulte el código origen de Ion simple.

En las siguientes secciones se muestra cómo ejecutar operaciones CRUD básicas, especificar una lógica de reintento personalizada e implementar restricciones de exclusividad.

Contenido

Creación de tablas

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

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

Creación de índices

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

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

Lectura de documentos

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

def read_documents(transaction_executor):
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId =
 'TOYENC486FH'")

    for doc in cursor:
```

```
        print(doc["GovId"]) # prints TOYENC486FH
        print(doc["FirstName"]) # prints Brent

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

Uso de parámetros de consulta

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo nativo.

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

En el siguiente ejemplo de código, se utiliza un parámetro de consulta de tipo Ion.

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

En el siguiente ejemplo de código se utilizan varios parámetros de consulta.

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

En el siguiente ejemplo de código, se utiliza una lista de parámetros de consulta.

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

> **Note**
>
> Cuando ejecuta una consulta sin una búsqueda indexada, se invoca un escaneo completo de la tabla. En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las consultas pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Inserción de documentos

El siguiente ejemplo de código inserta los tipos de datos nativos.

```
def insert_documents(transaction_executor, arg_1):
    # Check if doc with GovId:TOYENC486FH exists
    # This is critical to make this transaction idempotent
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", 'TOYENC486FH')
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", arg_1)

doc_1 = { 'FirstName': "Brent",
          'GovId': 'TOYENC486FH',
        }

qldb_driver.execute_lambda(lambda executor: insert_documents(executor, doc_1))
```

El siguiente ejemplo de código inserta los tipos de datos de Ion.

```
def insert_documents(transaction_executor, arg_1):
    # Check if doc with GovId:TOYENC486FH exists
    # This is critical to make this transaction idempotent
    cursor = transaction_executor.execute_statement("SELECT * FROM Person WHERE GovId
 = ?", 'TOYENC486FH')
    # Check if there is any record in the cursor
    first_record = next(cursor, None)

    if first_record:
        # Record already exists, no need to insert
        pass
    else:
        transaction_executor.execute_statement("INSERT INTO Person ?", arg_1)

doc_1 = { 'FirstName': 'Brent',
          'GovId': 'TOYENC486FH',
        }

# create a sample Ion doc
ion_doc_1 = simpleion.loads(simpleion.dumps(doc_1)))
```

```
qldb_driver.execute_lambda(lambda executor: insert_documents(executor, ion_doc_1))
```

Esta transacción inserta un documento en la tabla `Person`. Antes de insertar, compruebe primero si el documento ya existe en la tabla. Esta comprobación hace que la transacción sea de naturaleza idempotente. Incluso si realiza esta transacción varias veces, no provocará ningún efecto secundario no deseado.

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Insertar varios documentos en una instrucción

Para insertar varios documentos mediante una sola instrucción INSERT, puede pasar un parámetro del tipo list a la instrucción de la siguiente manera.

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

No coloque el marcador de posición variable (?) entre corchetes de doble ángulo (`<<...>>`) al pasar una lista. En las instrucciones PartiQL manuales, los corchetes de doble ángulo indican una colección desordenada conocida como bolsa.

Actualización de documentos

El siguiente ejemplo de código utiliza tipos de datos nativos.

```
def update_documents(transaction_executor, gov_id, name):
    transaction_executor.execute_statement("UPDATE Person SET FirstName = ?  WHERE
 GovId = ?", name, gov_id)

gov_id = 'TOYENC486FH'
name = 'John'

qldb_driver.execute_lambda(lambda executor: update_documents(executor, gov_id, name))
```

El siguiente ejemplo de código utiliza tipos de datos Ion.

```
def update_documents(transaction_executor, gov_id, name):
    transaction_executor.execute_statement("UPDATE Person SET FirstName = ? WHERE GovId
 = ?", name, gov_id)

# Ion datatypes
gov_id = simpleion.loads('TOYENC486FH')
name = simpleion.loads('John')

qldb_driver.execute_lambda(lambda executor: update_documents(executor, gov_id, name))
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Eliminación de documentos

El siguiente ejemplo de código utiliza tipos de datos nativos.

```
def delete_documents(transaction_executor, gov_id):
    cursor = transaction_executor.execute_statement("DELETE FROM Person WHERE GovId
 = ?", gov_id)

gov_id = 'TOYENC486FH'

qldb_driver.execute_lambda(lambda executor: delete_documents(executor, gov_id))
```

El siguiente ejemplo de código utiliza tipos de datos Ion.

```
def delete_documents(transaction_executor, gov_id):
    cursor = transaction_executor.execute_statement("DELETE FROM Person WHERE GovId
 = ?", gov_id)

# Ion datatypes
gov_id = simpleion.loads('TOYENC486FH')

qldb_driver.execute_lambda(lambda executor: delete_documents(executor, gov_id))
```

> **Note**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Ejecutar varias instrucciones en una transacción

```
# This code snippet is intentionally trivial. In reality you wouldn't do this because
 you'd
# set your UPDATE to filter on vin and insured, and check if you updated something or
 not.

def do_insure_car(transaction_executor, vin):
    cursor = transaction_executor.execute_statement(
        "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
    first_record = next(cursor, None)
    if first_record:
        transaction_executor.execute_statement(
            "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
        return True
    else:
        return False

def insure_car(qldb_driver, vin_to_insure):
    return qldb_driver.execute_lambda(
        lambda executor: do_insure_car(executor, vin_to_insure))
```

Lógica de reintentos

El método del controlador `execute_lambda` tiene un mecanismo de reintento integrado que reintenta la transacción si se produce una excepción que se pueda volver a intentar (como tiempos de espera o conflictos de OCC).

3.x

El número máximo de reintentos y la estrategia de retardo se pueden configurar.

El límite de reintentos predeterminado es `4` y la estrategia de retardo predeterminada es retroceso exponencial y fluctuación, con una base de `10` milisegundos. Puede establecer la configuración

de reintentos por instancia de controlador y también por transacción mediante una instancia de pyqldb.config.retry_config.RetryConfig.

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una instancia del controlador.

```
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)
```

El siguiente ejemplo de código especifica la lógica de reintentos con un límite de reintentos personalizado y una estrategia de retardo personalizada para una ejecución lambda particular. Esta configuración `execute_lambda` anula la lógica de reintentos establecida para la instancia del controlador.

```
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

El número máximo de reintentos se puede configurar. Puede configurar el límite de reintentos estableciendo la propiedad `retry_limit` al inicializar `PooledQldbDriver`.

El límite de reintentos predeterminado es de 4.

Implementación de restricciones de exclusividad

QLDB no admite índices únicos, pero puede implementar este comportamiento en su aplicación.

Suponga que desea implementar una restricción de exclusividad en el campo `GovId` de la tabla `Person`. Para ello, puede escribir una transacción que haga lo siguiente:

1. Afirmar que en la tabla no hay documentos existentes con un `GovId` especificado.
2. Insertar el documento si se aprueba la afirmación.

Si una transacción competidora supera la afirmación simultáneamente, solo una de las transacciones se confirmará correctamente. La otra transacción fallará y se producirá una excepción de conflicto de OCC.

En el siguiente ejemplo de código, se muestra cómo implementar esta lógica de restricción de exclusividad.

```
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**
>
> En este ejemplo, se recomienda tener un índice en el campo `GovId` para optimizar el rendimiento. Sin un índice en `GovId`, las instrucciones pueden tener más latencia y, además, provocar excepciones de conflictos de OCC o tiempos de espera de las transacciones.

Trabajar con Amazon Ion

Las siguientes secciones muestran cómo usar el módulo Amazon Ion para procesar los datos de Ion.

Contenido

Importación del módulo Ion

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

Creación de tipos de Ion

El siguiente ejemplo de código crea un objeto Ion a partir del texto de Ion.

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

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

El siguiente ejemplo de código crea un objeto Ion a partir de un `dict` de Python.

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

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

Obtener un volcado binario de Ion

```
# ion_obj is an Ion struct
print(simpleion.dumps(ion_obj)) # b'\xe0\x01\x00\xea\xee\x97\x81\x83\xde\x93\x87\xbe
\x90\x85GovId\x89FirstName\xde\x94\x8a\x8bTOYENC486FH\x8b\x85Brent'
```

Obtener un volcado de texto de Ion

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

Para obtener más información acerca de cómo trabajar con Ion, consulte la documentación de Amazon Ion en GitHub. Para ver más ejemplos de código sobre cómo trabajar con Ion en QLDB, consulte Uso de tipos de datos de Amazon Ion en Amazon QLDB.

Descripción de la gestión de sesiones con el controlador en Amazon QLDB

Si tiene experiencia en el uso de un sistema de administración de base de datos relacional (RDBMS), puede que esté familiarizado con las conexiones simultáneas. QLDB no tiene el mismo concepto de conexión RDBMS tradicional, ya que las transacciones se ejecutan con mensajes de solicitud y respuesta HTTP.

El concepto análogo en QLDB es el de sesión activa. Conceptualmente, una sesión es similar al inicio de sesión de un usuario: gestiona la información sobre sus solicitudes de transacción de datos a un libro mayor. Una sesión activa es aquella en la que se ejecuta una transacción de forma activa. También puede tratarse de una sesión en la que se ha finalizado recientemente una transacción, y el servicio prevé iniciar otra transacción de forma inmediata. QLDB admite una transacción en ejecución activa por sesión.

El límite de sesiones activas simultáneas por libro mayor se define en Cuotas y límites de Amazon QLDB. Una vez alcanzado este límite, cualquier sesión que intente iniciar una transacción dará como resultado un error (`LimitExceededException`).

Para conocer las prácticas recomendadas para configurar un grupo de sesiones en su aplicación mediante el controlador QLDB, consulte Configuración del objeto QLDBDriver en las Recomendaciones de controlador Amazon QLDB.

Contenido

Ciclo de vida de la sesión

La siguiente secuencia de operaciones de la API de sesión de QLDB representa el ciclo de vida típico de una sesión de QLDB:

1. `StartSession`
2. `StartTransaction`
3. `ExecuteStatement`
4. `CommitTransaction`
5. Repita los pasos 2 a 4 para iniciar más transacciones (una transacción a la vez).
6. `EndSession`

Vencimiento de la sesión

QLDB finaliza y descarta una sesión tras una duración total de 13 a 17 minutos, independientemente de si se está ejecutando una transacción de forma activa. Las sesiones se pueden perder o invalidar por distintos motivos, como fallos de hardware, fallos de red o reinicio de las aplicaciones. QLDB impone una duración máxima a las sesiones para garantizar que la aplicación cliente sea resiliente a fallos de sesión.

Gestión de sesiones en el controlador QLDB

Si bien puede usar un SDK de AWS para interactuar directamente con la API de sesión de QLDB, este paso añade complejidad y requiere el cálculo de un resumen de confirmación. QLDB emplea este resumen de confirmación para garantizar la integridad de la transacción. En lugar de interactuar directamente con esta API, recomendamos usar el controlador QLDB.

El controlador proporciona una capa de abstracción de alto nivel sobre la API de datos transaccionales. Simplifica el proceso de ejecución de instrucciones PartiQL en los datos del libro mayor gestionando las llamadas a la API SendCommand. Estas llamadas a la API necesitan de varios parámetros que el controlador gestiona automáticamente, administrando las sesiones, transacciones y política de reintentos en caso de errores.

Temas

Información general de la agrupación de sesiones

En versiones anteriores del controlador QLDB (como Java v1.1.0), proporcionábamos dos implementaciones del objeto controlador: un `QldbDriver` estándar no agrupado y un `PooledQldbDriver`. Como su nombre indica, `PooledQldbDriver` mantiene un conjunto de sesiones que se reutilizan en las transacciones.

A tenor de los comentarios de los usuarios, los desarrolladores prefieren usar la función de agrupación y sus ventajas de forma predeterminada en lugar de usar el controlador estándar. Por tanto, eliminamos `PooledQldbDriver` y trasladamos la funcionalidad de agrupación de sesiones a `QldbDriver`. Este cambio se incluye en la última versión de cada controlador (por ejemplo, Java v2.0.0).

El controlador proporciona tres niveles de abstracciones:

- Controlador (implementación de controlador agrupado): abstracción de nivel superior. El controlador mantiene y gestiona un conjunto de sesiones. Cuando solicita la ejecución de una transacción al controlador, este elige una sesión del grupo y la usa para ejecutar la transacción. Si la transacción falla debido a un error de sesión (`InvalidSessionException`), el controlador

elige otra sesión para reintentar la transacción. Básicamente, el controlador ofrece una experiencia de sesión totalmente gestionada.

- Sesión: un nivel por debajo de la abstracción del controlador. La sesión está incluida en un grupo, y el controlador gestiona el ciclo de vida de la sesión. Si se produce un error en una transacción, el controlador lleva a cabo un número específico de reintentos en la misma sesión. Si la sesión devuelve un error (`InvalidSessionException`), QLDB la descarta internamente. A continuación, el controlador asigna otra sesión del grupo para reintentar la transacción.
- Transacción: el nivel más bajo de abstracción. Una transacción está contenida en una sesión, y la sesión gestiona el ciclo de vida de la transacción. La sesión reintenta la transacción en caso de error. La sesión también garantiza que no se filtre una transacción abierta que no se haya confirmado o cancelado.

En la versión más reciente de cada controlador, solo es posible realizar operaciones en el nivel de abstracción de controlador. El usuario no tiene control directo sobre las sesiones y transacciones individuales (es decir, no hay operaciones de API que permitan iniciar manualmente una nueva sesión o transacción).

Agrupación de sesiones y lógica de transacciones

La última versión de cada controlador ya no proporciona una implementación no agrupada del objeto de controlador. De forma predeterminada, el objeto `QldbDriver` administra el grupo de sesiones. Al realizar una llamada para ejecutar una transacción, el controlador sigue estos pasos:

1. El controlador comprueba si se ha alcanzado el límite de sesiones acumuladas. Si es así, el controlador lanza de inmediato una excepción `NoSessionAvailable`. De lo contrario, continúa con el próximo paso.
2. El controlador comprueba si el grupo tiene una sesión disponible.
   - Si hay una sesión disponible en el grupo, el controlador la usa para ejecutar una transacción.
   - Si no hay una sesión disponible en el grupo, el controlador crea una nueva sesión y la usa para ejecutar una transacción.
3. Cuando el controlador asigna una sesión en el paso 2, realiza una llamada a la operación `execute` en la instancia de sesión.
4. En la operación `execute` de la sesión, el controlador intenta iniciar una transacción mediante una llamada `startTransaction`.

- Si la sesión no es válida, la llamada `startTransaction` falla y el controlador regresa al paso 1.
- Si la llamada `startTransaction` se realiza correctamente, el controlador continúa con el siguiente paso.

5. El controlador ejecuta la expresión lambda. Esta expresión lambda puede contener una o más llamadas para ejecutar instrucciones PartiQL. Cuando la expresión lambda finaliza su ejecución sin errores, el controlador procede a confirmar la transacción.
6. La confirmación de la transacción puede devolver uno de estos dos resultados:
   - La confirmación se realiza correctamente y el controlador devuelve el control al código de la aplicación.
   - La confirmación falla debido a un conflicto de control de concurrencia optimista (OCC). En este caso, el controlador reintenta los pasos 4 a 6 usando la misma sesión. Puede configurar el número máximo de reintentos en el código de su aplicación. El límite predeterminado es 4.

> **Note**
>
> Si se devuelve `InvalidSessionException` durante los pasos 4 a 6, el controlador marca la sesión como cerrada y vuelve al paso 1 para reintentar la transacción.
> Si se produce alguna otra excepción durante los pasos 4 a 6, el controlador comprueba si se puede reintentar la excepción. Si es así, reintenta la transacción hasta el número especificado de reintentos. De lo contrario, propaga la excepción al código de la aplicación.

Devolución de las sesiones al grupo

Si la transacción activa devuelve `InvalidSessionException` en cualquier momento, el controlador no devuelve la sesión al grupo. En su lugar, QLDB descarta la sesión y el controlador asigna otra sesión del grupo. En el resto de casos, el controlador devuelve la sesión al grupo.

Recomendaciones de controladores de Amazon QLDB

En esta sección se describen las prácticas recomendadas para configurar y utilizar el controlador Amazon QLDB para cualquier lenguaje compatible. Los ejemplos de código proporcionados son específicos para Java.

Estas recomendaciones se aplican a la mayoría de los casos de uso típicos, pero no hay una solución única para todos. Utilice las siguientes recomendaciones como mejor le parezca para su aplicación.

Temas

Configuración del objeto QLDBDriver

El objeto `QldbDriver` administras las conexiones a su libro mayor manteniendo un conjunto de sesiones que se reutilizan en todas las transacciones. Una sesión representa una conexión única con el libro mayor. QLDB admite una transacción en ejecución activa por sesión.

Important

En las versiones anteriores del controlador, la funcionalidad de agrupación de sesiones sigue estando en el objeto `PooledQldbDriver` y no en `QldbDriver`. Si utiliza una de las siguientes versiones, sustituya cualquier mención de `QldbDriver` por `PooledQldbDriver` para el resto de este tema.

Controlador	Versión
Java	`1.1.0` o anterior
.NET	`0.1.0-beta`
Node.js	`1.0.0-rc.1` o anterior
Python	`2.0.2` o anterior

El objeto `PooledQldbDriver` está obsoleto en la versión más reciente de los controladores. Se recomienda actualizar a la última versión y convertir cualquier instancia de `PooledQldbDriver` a `QldbDriver`.

Configurar QLDBDriver como un objeto global

Para optimizar el uso de los controladores y las sesiones, asegúrese de que solo exista una instancia global del controlador en la instancia de la aplicación. Por ejemplo, en Java, puede usar marcos de inyección de dependencias como Spring, Google Guice o Dagger. En el ejemplo de código siguiente se muestra cómo configurar `QldbDriver` como singleton.

```
@Singleton
public QldbDriver qldbDriver (AWSCredentialsProvider credentialsProvider,
                                    @Named(LEDGER_NAME_CONFIG_PARAM) String ledgerName)
 {
    QldbSessionClientBuilder builder = QldbSessionClient.builder();
    if (null != credentialsProvider) {
        builder.credentialsProvider(credentialsProvider);
    }
    return QldbDriver.builder()
            .ledger(ledgerName)
            .transactionRetryPolicy(RetryPolicy
                .builder()
                .maxRetries(3)
                .build())
            .sessionClientBuilder(builder)
            .build();
}
```

Configurar los reintentos

El controlador vuelve a intentar las transacciones automáticamente cuando se producen excepciones transitorias comunes (por ejemplo, `SocketTimeoutException` o `NoHttpResponseException`). Para establecer el número máximo de reintentos, puede utilizar el parámetro `maxRetries` del objeto de configuración `transactionRetryPolicy` al crear una instancia de `QldbDriver`. (Para versiones anteriores del controlador, como se indica en la sección anterior, utilice el parámetro `retryLimit` de `PooledQldbDriver`.)

El valor predeterminado de `maxRetries` es `4`.

Errores del lado del cliente, como `InvalidParameterException` no se pueden volver a intentar. Cuando se producen, la transacción se cancela, la sesión se devuelve al grupo y la excepción se envía al cliente del controlador.

Configurar el número máximo de sesiones y transacciones simultáneas

El número máximo de sesiones de libro mayor que utiliza una instancia `QldbDriver` para ejecutar transacciones viene definido por su parámetro `maxConcurrentTransactions`. (Para versiones anteriores del controlador, como se indica en la sección anterior, se define en el parámetro `poolLimit` de `PooledQldbDriver`.)

Este límite debe ser superior a cero e inferior o igual al número máximo de conexiones HTTP abiertas que permite el cliente de sesión, según lo definido en el SDK de AWS específico. Por ejemplo, en Java, el número máximo de conexiones se establece en el objeto ClientConfiguration.

El valor predeterminado de `maxConcurrentTransactions` es la configuración de conexión máxima de su SDK de AWS.

Cuando configure `QldbDriver` en su aplicación, tenga en cuenta las siguientes consideraciones de escalado:

- Su grupo siempre debe tener al menos tantas sesiones como el número de transacciones en ejecución simultánea que planea tener.
- En un modelo de subprocesos múltiples en el que un subproceso supervisor delega en subprocesos de trabajo, el controlador debe tener al menos tantas sesiones como el número de subprocesos de trabajo. De lo contrario, en el momento de máxima carga, los subprocesos estarán esperando en fila hasta que haya una sesión disponible.
- El límite de servicio de sesiones activas simultáneas por libro mayor se define en Cuotas y límites de Amazon QLDB. Asegúrese de no haber configurado un número de sesiones simultáneas superior a este límite para utilizarlas en un solo libro mayor en todos los clientes.

Reintentar en caso de excepciones

Al volver a intentar las excepciones que se producen en la QLDB, tenga en cuenta las siguientes recomendaciones.

Reintentar en caso de OccConflictException

Las excepciones de conflicto relacionadas con el control de concurrencia optimista (OCC) se producen cuando los datos a los que accede la transacción han cambiado desde el inicio de la transacción. QLDB lanza esta excepción mientras intenta confirmar la transacción. El controlador vuelve a intentar la transacción tantas veces como se haya configurado en `maxRetries`.

Para obtener más información sobre la OCC y las prácticas recomendadas para utilizar índices para limitar los conflictos de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Reintentar con otras excepciones fuera de QldbDriver

Para reintentar una transacción fuera del controlador cuando se producen excepciones personalizadas y definidas por la aplicación durante el tiempo de ejecución, debe empaquetar la transacción. Por ejemplo, en Java, el código siguiente muestra cómo utilizar la biblioteca Reslience4J para reintentar una transacción en QLDB.

```
private final RetryConfig retryConfig = RetryConfig.custom()
        .maxAttempts(MAX_RETRIES)
        .intervalFunction(IntervalFunction.ofExponentialRandomBackoff())
        // Retry this exception
        .retryExceptions(InvalidSessionException.class, MyRetryableException.class)
        // But fail for any other type of exception extended from RuntimeException
        .ignoreExceptions(RuntimeException.class)
        .build();

// Method callable by a client
public void myTransactionWithRetries(Params params) {
    Retry retry = Retry.of("registerDriver", retryConfig);

    Function<Params, Void> transactionFunction = Retry.decorateFunction(
            retry,
            parameters ->  transactionNoReturn(params));
    transactionFunction.apply(params);
}

private Void transactionNoReturn(Params params) {
    try (driver.execute(txn -> {
            // Transaction code
        });
    }
    return null;
}
```

> **Note**
>
> Reintentar una transacción fuera del controlador QLDB tiene un efecto multiplicador. Por ejemplo, si `QldbDriver` está configurado para volver a intentarlo tres veces y la lógica de reintento personalizada también lo hace tres veces, se puede volver a intentar la misma transacción hasta nueve veces.

Hacer que las transacciones sean idempotentes

Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos. Una transacción es idempotente si puede ejecutarse varias veces y producir resultados idénticos cada vez.

Para obtener más información, consulte Modelo de concurrencia de Amazon QLDB.

Optimización del rendimiento

Para optimizar el rendimiento al ejecutar transacciones con el controlador, tenga en cuenta las siguientes consideraciones:

- La operación `execute` siempre realiza un mínimo de tres llamadas a la API `SendCommand` a QLDB, incluidos los siguientes comandos:
  1. `StartTransaction`
  2. `ExecuteStatement`

     Este comando se invoca para cada instrucción PartiQL que ejecute en el bloque `execute`.
  3. `CommitTransaction`

  Tenga en cuenta el número total de llamadas a la API que se realizan al calcular la carga de trabajo total de la aplicación.
- En general, se recomienda empezar con un escritor de un solo subproceso y optimizar las transacciones agrupando varias instrucciones en una sola transacción. Maximice las cuotas de tamaño de transacción, tamaño de documento y cantidad de documentos por transacción, tal y como se define en Cuotas y límites de Amazon QLDB.
- Si el procesamiento por lotes no es suficiente para grandes cargas de transacciones, puede probar con varios subprocesos añadiendo instancias de escritura adicionales. Sin embargo, debe considerar detenidamente los requisitos de su solicitud para la secuenciación de documentos y transacciones y la complejidad adicional que esto implica.

Ejecutar varias instrucciones por transacción

Como se describe en la sección anterior, puede ejecutar varias instrucciones por transacción para optimizar el rendimiento de su aplicación. En el siguiente ejemplo de código, se consulta una tabla y, a continuación, se actualiza un documento de esa tabla dentro de una transacción. Para ello, debe pasar una expresión lambda a la operación `execute`.

Java

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

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

.NET

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
public static async Task<bool> InsureVehicle(IAsyncQldbDriver driver, string vin)
{
    ValueFactory valueFactory = new ValueFactory();
    IIonValue ionVin = valueFactory.NewString(vin);

    return await driver.Execute(async txn =>
    {
        // Check if the vehicle is insured.
        Amazon.QLDB.Driver.IAsyncResult result = await txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 ionVin);

        if (await result.CountAsync() > 0)
        {
            // If the vehicle is not insured, insure it.
```

```
            await txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", ionVin);
            return true;
        }
        return false;
    });
}
```

Go

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
func InsureCar(driver *qldbdriver.QLDBDriver, vin string) (bool, error) {
    insured, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {

        result, err := txn.Execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
        if err != nil {
            return false, err
        }

        hasNext := result.Next(txn)
        if !hasNext && result.Err() != nil {
            return false, result.Err()
        }

        if hasNext {
            _, err = txn.Execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
            if err != nil {
                return false, err
            }
            return true, nil
        }
        return false, nil
    })
    if err != nil {
        panic(err)
    }
```

```
    return insured.(bool), err
}
```

Node.js

```
// This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
// set your UPDATE to filter on vin and insured, and check if you updated something
 or not.
async function insureCar(driver: QldbDriver, vin: string): Promise<boolean> {

    return await driver.executeLambda(async (txn: TransactionExecutor) => {
        const results: dom.Value[] = (await txn.execute(
            "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE",
 vin)).getResultList();

        if (results.length > 0) {
            await txn.execute(
                "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin);
            return true;
        }
        return false;
    });
};
```

Python

```
# This code snippet is intentionally trivial. In reality you wouldn't do this
 because you'd
# set your UPDATE to filter on vin and insured, and check if you updated something
 or not.

def do_insure_car(transaction_executor, vin):
    cursor = transaction_executor.execute_statement(
        "SELECT insured FROM Vehicles WHERE vin = ? AND insured = FALSE", vin)
    first_record = next(cursor, None)
    if first_record:
        transaction_executor.execute_statement(
            "UPDATE Vehicles SET insured = TRUE WHERE vin = ?", vin)
        return True
    else:
        return False
```

```
def insure_car(qldb_driver, vin_to_insure):
    return qldb_driver.execute_lambda(
        lambda executor: do_insure_car(executor, vin_to_insure))
```

La operación `execute` del controlador inicia implícitamente una sesión y una transacción en esa sesión. Cada instrucción que se ejecuta en la expresión lambda se incluye en la transacción. Una vez ejecutadas todas las instrucciones, el controlador confirma automáticamente la transacción. Si alguna instrucción falla una vez agotado el límite de reintentos automáticos, la transacción se cancela.

Propagar las excepciones en una transacción

Cuando se ejecutan varias instrucciones por transacción, por lo general, no recomendamos detectar excepciones dentro de la transacción.

Por ejemplo, en Java, el siguiente programa detecta cualquier instancia de `RuntimeException`, registra el error y continúa. Este ejemplo de código se considera una mala práctica porque la transacción se realiza correctamente incluso cuando la instrucción UPDATE falla. Por lo tanto, el cliente podría suponer que la actualización se realizó correctamente, pero no fue así.

> **Warning**
>
> No utilice este ejemplo de código. Se proporciona para mostrar un ejemplo anti-patrón que se considera una mala práctica.

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

En su lugar, propague (haga crecer) la excepción. Si alguna parte de la transacción falla, deje que la operación `execute` cancele la transacción para que el cliente pueda gestionar la excepción en consecuencia.

Descripción de la política de reintentos del controlador en Amazon QLDB

El controlador de Amazon QLDB emplea una política de reintentos para gestionar las excepciones transitorias reintentando de forma transparente una transacción fallida. Estas excepciones, como `CapacityExceededException` y `RateExceededException`, suelen corregirse por sí solas tras un determinado período de tiempo. Si la transacción que falló con la excepción se reintenta tras un lapso adecuado, es probable que se realice correctamente. Esto ayuda a mejorar la estabilidad de la aplicación que usa QLDB.

Temas

- Tipos de errores que se pueden reintentar
- Política de reintento predeterminada

Tipos de errores que se pueden reintentar

El controlador reintenta una transacción automáticamente si, y solo si, se produce alguna de las siguientes excepciones durante una operación dentro de dicha transacción:

- CapacityExceededException: se devuelve cuando la solicitud supera la capacidad de procesamiento del libro mayor.
- InvalidSessionException: se devuelve cuando una sesión ya no es válida o si la sesión no existe.

- LimitExceededException: se devuelve si se supera un límite de recursos, como el número de sesiones activas.
- OccConflictException: se devuelve cuando no se puede escribir una transacción en el diario debido a un fallo en la fase de verificación del control de concurrencia optimista (OCC).
- RateExceedException: se devuelve cuando la tasa de solicitudes supera el rendimiento permitido.

Política de reintento predeterminada

La política de reintentos consta de una condición de reintento y una estrategia de espera. La condición de reintento define cuándo se debe volver a intentar una transacción, mientras que la estrategia de espera define cuánto tiempo se debe esperar antes de reintentar la transacción.

Al crear una instancia del controlador, la política de reintentos predeterminada especifica hasta cuatro reintentos con una estrategia de espera de retroceso exponencial. La estrategia de retroceso exponencial emplea un retraso mínimo de 10 milisegundos y un retraso máximo de 5000 milisegundos, con la misma fluctuación de fase. Si la transacción no se puede confirmar correctamente en la política de reintentos, le recomendamos que trate de realizarla en otro momento.

El concepto de retroceso exponencial se basa en el concepto de utilizar tiempos de espera progresivamente más largos entre reintentos para las respuestas a errores consecutivos. Para obtener más información, consulte la entrada de blog AWSRetroceso exponencial y fluctuación.

Errores comunes del controlador QLDB de Amazon

En esta sección se describen los errores de tiempo de ejecución que puede generar el controlador de Amazon QLDB al interactuar con la API de sesión de QLDB.

La siguiente es una lista de excepciones comunes devueltas por el controlador. Cada excepción incluye el mensaje de error específico, seguido de una breve descripción y sugerencias de posibles soluciones.

CapacityExceededException

Mensaje: Capacidad superada

Amazon QLDB rechazó la solicitud porque superaba la capacidad de procesamiento del libro mayor. La QLDB impone un límite de escalado interno por libro mayor para mantener el estado y el rendimiento del servicio. Este límite varía según el tamaño de la carga de trabajo de cada solicitud individual. Por ejemplo, una solicitud puede tener una mayor carga de trabajo si realiza

transacciones de datos ineficientes, como los escaneos de tablas que resultan de una consulta no apta para indexar.

Le recomendamos que espere antes de volver a intentar la solicitud. Si su solicitud encuentra esta excepción de forma constante, optimice sus instrucciones y reduzca la frecuencia y el volumen de las solicitudes que envía al libro mayor. Algunos ejemplos de optimización de instrucciones incluyen ejecutar menos instrucciones por transacción y ajustar los índices de las tablas. Para obtener información sobre cómo optimizar las instrucciones y evitar el escaneo de tablas, consulte Optimizar el rendimiento de las consultas.

Siempre recomendamos usar la versión más reciente del controlador QLDB. El controlador tiene una política de reintentos predeterminada que utiliza el Retroceso exponencial y la fluctuación de fase para volver a intentarlo automáticamente en excepciones como esta. El concepto de retroceso exponencial se basa en utilizar tiempos de espera progresivamente más largos entre reintentos para las respuestas a errores consecutivos.

InvalidSessionException

Mensaje: Transaction *`transactionId`* has expired

Una transacción ha superado su vida útil máxima. Una transacción puede ejecutarse durante un máximo de 30 segundos antes de confirmarse. Tras este límite de tiempo de espera, se rechaza cualquier trabajo realizado en la transacción y la QLDB descarta la sesión. Este límite evita que el cliente pierda sesiones al iniciar transacciones y no confirmarlas ni cancelarlas.

Si se trata de una excepción habitual en su aplicación, es probable que las transacciones simplemente estén tardando demasiado en ejecutarse. Si el tiempo de ejecución de la transacción supera los 30 segundos, optimice sus instrucciones para acelerar las transacciones. Algunos ejemplos de optimización de instrucciones incluyen ejecutar menos instrucciones por transacción y ajustar los índices de las tablas. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

InvalidSessionException

Mensaje: Session *`sessionId`* has expired

La QLDB descartó la sesión porque excedió su vida útil total máxima. La QLDB descarta las sesiones después de 13 a 17 minutos, independientemente de si hay una transacción activa. Las sesiones se pueden perder o dañar por distintos motivos, como fallos de hardware, fallos de red o reinicio de las aplicaciones. QLDB impone una duración máxima a las sesiones para garantizar que el software cliente sea resiliente a fallos de sesión.

Si se produce esta excepción, le recomendamos que inicie una sesión nueva y vuelva a intentar la transacción. También recomendamos utilizar la última versión del controlador QLDB, que administra el grupo de sesiones y su estado en nombre de la aplicación.

InvalidSessionException

Mensaje: No such session

El cliente intentó realizar transacciones con QLDB mediante una sesión que no existe. Suponiendo que el cliente esté utilizando una sesión que existía anteriormente, es posible que la sesión ya no exista debido a uno de los siguientes motivos:

- Si una sesión está involucrada en un fallo interno del servidor (es decir, un error con el código de respuesta HTTP 500), QLDB podría optar por descartar la sesión por completo, en lugar de permitir que el cliente realice transacciones con una sesión de estado incierto. En ese caso, cualquier intento de reintento en esa sesión fallará y se generará este error.
- QLDB finalmente olvida las sesiones caducadas. De ahí que, cualquier intento de continuar usando la sesión produzca este error, en lugar del `InvalidSessionException` inicial.

Si se produce esta excepción, le recomendamos que inicie una sesión nueva y vuelva a intentar la transacción. También recomendamos utilizar la última versión del controlador QLDB, que administra el grupo de sesiones y su estado en nombre de la aplicación.

RateExceededException

Mensaje: The rate was exceeded

QLDB limitó a un cliente en función de la identidad de la persona que llamaba. QLDB impone la limitación por región y por cuenta mediante un algoritmo de limitación de bucket de token. Lo hace para mejorar el rendimiento del servicio y garantizar un utilización justa para todos los clientes de QLDB. Por ejemplo, intentar adquirir una gran cantidad de sesiones simultáneas mediante la operación `StartSessionRequest` podría provocar una limitación.

Para mantener el buen estado de la aplicación y reducir aún más las limitaciones, puede volver a intentarlo con esta excepción mediante el Retroceso exponencial y la fluctuación de fase. El concepto de retroceso exponencial se basa en utilizar tiempos de espera progresivamente más largos entre reintentos para las respuestas a errores consecutivos. Siempre recomendamos usar la versión más reciente del controlador QLDB. El controlador tiene una política de reintentos predeterminada que utiliza el Retroceso exponencial y la fluctuación de fase para volver a intentarlo automáticamente en excepciones como esta.

La última versión del controlador QLDB también puede ayudar si QLDB limita constantemente su aplicación para llamadas `StartSessionRequest`. El controlador mantiene un conjunto de sesiones que se reutilizan en todas las transacciones, lo que puede ayudar a reducir el número de llamadas `StartSessionRequest` que realiza su aplicación. Para solicitar un aumento de los límites de la limitación API, contacte con el Centro AWS Support.

LimitExceededException

Mensaje: Exceeded the session limit

Un libro mayor ha superado su cuota (también conocida como límite) en cuanto al número de sesiones activas. Esta cuota se define en Cuotas y límites de Amazon QLDB. Con el tiempo, el recuento de sesiones activas de un libro mayor es uniforme, y los libros que se acerquen constantemente a la cuota podrían ver esta excepción periódicamente.

Para mantener el buen estado de la aplicación, le recomendamos que vuelva a intentarlo con esta excepción. Para evitar esta excepción, asegúrese de no haber configurado más de 1500 sesiones simultáneas para utilizarlas en un solo libro mayor en todos los clientes. Por ejemplo, puede usar el método maxConcurrentTransactions del controlador Amazon QLDB para Java para configurar el número máximo de sesiones disponibles en una instancia de controlador.

QldbClientException

Mensaje: A streamed result is only valid when the parent transaction is open

La transacción está cerrada y no se puede utilizar para recuperar los resultados de la QLDB. Una transacción se cierra cuando se confirma o se cancela.

Esta excepción se produce cuando el cliente trabaja directamente con el objeto `Transaction` e intenta recuperar los resultados de la QLDB después de haber confirmado o cancelado una transacción. Para mitigar este problema, el cliente debe leer los datos antes de cerrar la transacción.

Tutorial de introducción a Amazon QLDB mediante un ejemplo de aplicación

En este tutorial, utilizará el controlador Amazon QLDB con el SDK de AWS para crear un libro mayor de QLDB y rellenarlo con datos de ejemplo. El controlador permite que su aplicación interactúe con QLDB mediante la API de datos transaccionales. El SDK de AWS admite la interacción con la API de administración de recursos de QLDB.

El libro mayor que crea en este ejemplo es una base de datos del departamento de vehículos automóviles (DMV) que rastrea la información histórica completa de las matriculaciones de vehículos. En los siguientes temas se explica cómo agregar registros de vehículos, modificarlos y ver el historial de cambios en esos registros. En esta guía también se muestra cómo verificar criptográficamente un documento de registro y, por último, se limpian los recursos y se elimina el ejemplo del libro mayor.

Este tutorial de una aplicación de muestra está disponible para los lenguajes de programación que se indican a continuación.

Temas

- Tutorial de Amazon QLDB para Java
- Tutorial de Amazon QLDB Node.js
- Tutorial de Python para Amazon QLDB

Tutorial de Amazon QLDB para Java

En esta implementación de la aplicación de ejemplo del tutorial, utilizará el controlador Amazon QLDB con AWS SDK for Java para crear un libro mayor de QLDB y rellenarlo con datos de ejemplo.

Mientras realiza este tutorial, puede consultar la Referencia de la API de AWS SDK for Java. Para las operaciones de datos transaccionales, puede consultar el controlador QLDB para la referencia de la API de Java.

> **Note**
>
> Cuando proceda, algunos pasos del tutorial incluyen comandos o ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Java.

Temas

- Instalación de la aplicación de ejemplo Java de Amazon QLDB
- Paso 1: Crear un nuevo libro mayor
- Paso 2: para probar la conectividad con el libro mayor
- Paso 3: cree tablas, índices y datos de muestra
- Paso 4: consultar las tablas en un libro mayor
- Paso 5: modificar los documentos de un libro mayor

- Paso 6: ver el historial de revisiones de un documento
- Paso 7: verificar un documento en un libro mayor
- Paso 8: exportar y validar los datos del diario en un libro mayor
- Paso 9 (opcional): limpiar recursos

Instalación de la aplicación de ejemplo Java de Amazon QLDB

En esta sección se describe cómo instalar y ejecutar la aplicación de ejemplo de Amazon QLDB proporcionada para este tutorial de Java paso a paso. El caso de uso de esta aplicación de ejemplo es una base de datos del Departamento de Vehículos Automóviles (DMV) que rastrea la información histórica completa de las matriculaciones de vehículos.

Esta aplicación de ejemplo de DMV para Java es de código abierto y se encuentra en el repositorio de GitHub aws-samples/amazon-qldb-dmv-sample-java.

Requisitos previos

Antes de comenzar, asegúrese de completar el controlador QLDB para Java Requisitos previos. Estas incluyen las siguientes:

1. Regístrese en AWS.
2. Cree un usuario con los permisos de QLDB adecuados. Para completar todos los pasos de este tutorial, necesitará acceso administrativo completo a su recurso de libro mayor a través de la API de QLDB.
3. Si usa un IDE distinto a AWS Cloud9, instale Java y conceda acceso programático de desarrollo.

Instalación

Los siguientes pasos describen cómo descargar y configurar la aplicación de muestra en un entorno de desarrollo local. También puede automatizar la configuración de la aplicación de muestra usando AWS Cloud9 como IDE y aprovisionando los recursos de desarrollo con una plantilla de AWS CloudFormation.

Entorno de desarrollo local

Estas instrucciones describen cómo descargar e instalar la aplicación de ejemplo Java de QLDB usando sus propios recursos y entorno de desarrollo.

Para descargar y ejecutar la aplicación de muestra

1. Introduzca el siguiente comando para clonar la aplicación de muestra desde GitHub.

   2.x

   ```
   git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-java.git
   ```

   1.x

   ```
   git clone -b v1.2.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-
   java.git
   ```

   Este paquete incluye la configuración de Gradle y el código completo de Tutorial de Java.

2. Cargue y ejecute la aplicación proporcionada.

   - Si usa Eclipse:

     a. Inicie Eclipse y, en el menú de Eclipse, seleccione Archivo, Importar y, a continuación, Proyecto de Gradle existente.

     b. En el directorio raíz del proyecto, busque y seleccione el directorio de aplicación que contiene el archivo `build.gradle`. A continuación, seleccione Finalizar para usar la configuración predeterminada de Gradle para la importación.

     c. Puede ejecutar el programa `ListLedgers` a modo de ejemplo. Abra el menú contextual (clic derecho) del archivo `ListLedgers.java` y seleccione Ejecutar como aplicación Java.

   - Si usa IntelliJ:

     a. Inicie IntelliJ y, en el menú de IntelliJ, seleccione Archivo y, a continuación, Abrir.

     b. En el directorio raíz del proyecto, busque y seleccione el directorio de aplicación que contiene el archivo `build.gradle`. A continuación, seleccione Aceptar. Mantenga la configuración predeterminada y vuelva a seleccionar Aceptar.

     c. Puede ejecutar el programa `ListLedgers` a modo de ejemplo. Abra el menú contextual (clic con el botón secundario) para el archivo `ListLedgers.java`, y elija Ejecutar 'ListLedgers'.

3. Continúe a Paso 1: Crear un nuevo libro mayor para iniciar el tutorial y crear un libro mayor.

AWS Cloud9

Estas instrucciones describen cómo automatizar la configuración de la aplicación de ejemplo de registro de vehículos de Amazon QLDB para Java, usando AWS Cloud9 como IDE. En esta guía, usará una plantilla de AWS CloudFormation para aprovisionar sus recursos de desarrollo.

Para obtener más información sobre AWS Cloud9, consulte la AWS Cloud9Guía del usuario de . Para obtener más información sobre AWS CloudFormation, consulte la Guía del usuario de AWS CloudFormation.

Temas

- Parte 1: aprovisione sus recursos
- Parte 2: configure el IDE
- Parte 3: ejecute la aplicación de ejemplo QLDB DMV

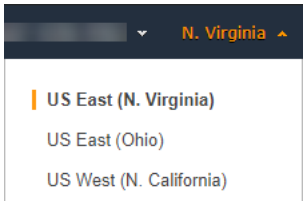
Parte 1: aprovisione sus recursos

En este primer paso, deberá usar AWS CloudFormation para aprovisionar los recursos necesarios para configurar su entorno de desarrollo con la aplicación de ejemplo de Amazon QLDB.

Para abrir la consola de AWS CloudFormation y cargar la plantilla de aplicación de ejemplo de QLDB

1. Inicie sesión en la AWS Management Console y abra la consola de AWS CloudFormation en https://console.aws.amazon.com/cloudformation.

   Cambie a una región compatible con QLDB. Para obtener una lista completa, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS. En la siguiente captura de pantalla de la AWS Management Console se muestra Este de EE. UU. (Norte de Virginia) como Región de AWS seleccionada.

2. En la consola de AWS CloudFormation, elija Crear pila, y, a continuación, elija Con nuevos recursos (estándar).
3. En la página Crear pila, en Especificar plantilla, elija URL de Amazon S3.
4. Introduzca la siguiente URL y seleccione Siguiente.

```
https://amazon-qldb-assets.s3.amazonaws.com/templates/QLDB-DMV-SampleApp.yml
```

5. Introduzca un nombre de pila (por ejemplo, **`qldb-sample-app`**) y elija Siguiente.
6. Puede añadir las etiquetas que desee y mantener las opciones predeterminadas. A continuación, elija Next.
7. Revise la configuración de su pila y seleccione Crear pila. El script de AWS CloudFormation podría tardar varios minutos en finalizar.

   Este script aprovisiona su entorno de AWS Cloud9 con una instancia de Amazon Elastic Compute Cloud (Amazon EC2) asociada que usará para ejecutar la aplicación de ejemplo de QLDB de este tutorial. También clona el repositorio aws-samples/amazon-qldb-dmv-sample-java de GitHub en su entorno de desarrollo AWS Cloud9.

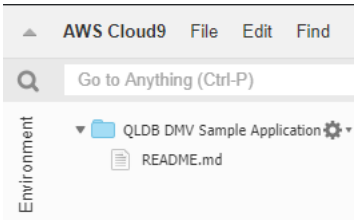
Parte 2: configure el IDE

En este paso, terminará de configurar el entorno de desarrollo en la nube. Descargue y ejecute el script de intérprete de comandos proporcionado para configurar su IDE de AWS Cloud9 con las dependencias de la aplicación de muestra.

Para configurar su entorno de AWS Cloud9

1. Abra la consola de AWS Cloud9 en https://console.aws.amazon.com/cloud9/.
2. En Sus entornos, busque la tarjeta de entorno denominada QLDB DMV Sample Application y seleccione Open IDE. Es posible que su entorno tarde un minuto en cargarse cuando se lance la instancia de EC2 subyacente.

   Su entorno de AWS Cloud9 está preconfigurado con las dependencias del sistema que necesita para ejecutar el tutorial. En el panel de navegación Entorno de la consola, confirme que ve una carpeta con el nombre `QLDB DMV Sample Application`. La siguiente captura de pantalla de la consola AWS Cloud9 muestra el panel de carpetas del entorno QLDB DMV Sample Application.



Si no ve ningún panel de navegación, active la pestaña Entorno, en la parte izquierda de la consola. Si no ve ninguna carpeta en el panel, active Mostrar raíz del entorno en el icono de configuración ().

3. En el panel inferior de la consola, debería ver una ventana de terminal de `bash` abierta. Si no lo ve, seleccione Nueva terminal en el menú Ventana de la parte superior de la consola.
4. A continuación, descargue y ejecute un script de configuración para instalar OpenJDK 8 y, si procede, consulte la ramificación correspondiente del repositorio de Git. En la terminal de AWS Cloud9 creada en el paso anterior, ejecute los dos siguientes comandos en este orden:

   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
   ```

   Al finalizar, debería ver el siguiente mensaje en la terminal:

   ```
   ** DMV Sample App setup completed , enjoy!! **
   ```

5. Tómese un momento para explorar el código de la aplicación de ejemplo en AWS Cloud9, especialmente la siguiente ruta de directorio: `src/main/java/software/amazon/qldb/tutorial`.

Parte 3: ejecute la aplicación de ejemplo QLDB DMV

En este paso, aprenderá a ejecutar las tareas de la aplicación de ejemplo DMV de Amazon QLDB usando AWS Cloud9. Para ejecutar el código de ejemplo, vuelva a la terminal de AWS Cloud9 o cree una nueva ventana de terminal, tal como hizo en la Parte 2: configure su IDE.

Para ejecutar las aplicaciones de ejemplo

1. Ejecute el siguiente comando en su terminal para pasar al directorio raíz del proyecto:

   ```
   cd ~/environment/amazon-qldb-dmv-sample-java
   ```

   Asegúrese de ejecutar los ejemplos en la siguiente ruta de directorio.

   ```
   /home/ec2-user/environment/amazon-qldb-dmv-sample-java/
   ```

2. El siguiente comando muestra la sintaxis de Gradle para ejecutar cada tarea.

   ```
   ./gradlew run -Dtutorial=Task
   ```

   Por ejemplo, ejecute el siguiente comando para ver todos los libros mayores de su Cuenta de AWS y región actuales.

   ```
   ./gradlew run -Dtutorial=ListLedgers
   ```

3. Continúe a Paso 1: Crear un nuevo libro mayor para iniciar el tutorial y crear un libro mayor.
4. (Opcional) Tras completar el tutorial, limpie los recursos de AWS CloudFormation si ya no los necesita.

   a. Abra la consola de AWS CloudFormation en https://console.aws.amazon.com/cloudformation y elimine la pila que creó en la Parte 1: aprovisione sus recursos.

   b. Elimine también la pila AWS Cloud9 creada por la plantilla de AWS CloudFormation.

Paso 1: Crear un nuevo libro mayor

En este paso, creará un nuevo libro de mayor de QLDB de Amazon denominado `vehicle-registration`.

Para crear un nuevo libro mayor

1. Revise el siguiente archivo (`Constants.java`), que contiene valores constantes que utilizan todos los demás programas de este tutorial.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonSystem;
import com.amazon.ion.system.IonSystemBuilder;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.dataformat.ion.IonObjectMapper;
import com.fasterxml.jackson.dataformat.ion.ionvalue.IonValueMapper;

/**
 * Constant values used throughout this tutorial.
 */
public final class Constants {
    public static final int RETRY_LIMIT = 4;
    public static final String LEDGER_NAME = "vehicle-registration";
    public static final String STREAM_NAME = "vehicle-registration-stream";
```

```
    public static final String VEHICLE_REGISTRATION_TABLE_NAME =
 "VehicleRegistration";
    public static final String VEHICLE_TABLE_NAME = "Vehicle";
    public static final String PERSON_TABLE_NAME = "Person";
    public static final String DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
    public static final String VIN_INDEX_NAME = "VIN";
    public static final String PERSON_GOV_ID_INDEX_NAME = "GovId";
    public static final String
 VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
    public static final String DRIVER_LICENSE_NUMBER_INDEX_NAME =
 "LicenseNumber";
    public static final String DRIVER_LICENSE_PERSONID_INDEX_NAME = "PersonId";
    public static final String JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-
tutorial-journal-export";
    public static final String USER_TABLES = "information_schema.user_tables";
    public static final String LEDGER_NAME_WITH_TAGS = "tags";
    public static final IonSystem SYSTEM = IonSystemBuilder.standard().build();
    public static final IonObjectMapper MAPPER = new IonValueMapper(SYSTEM);

    private Constants() { }

    static {
        MAPPER.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);
    }
}
```

1.x

> **Important**
>
> Para el paquete Amazon Ion, debe usar el espacio de nombres `com.amazon.ion` de su aplicación. AWS SDK for Java depende de otro paquete de Ion en el espacio de nombres `software.amazon.ion`, pero se trata de un paquete heredado que no es compatible con el controlador QLDB.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> Esta clase `Constants` incluye una instancia de la clase `IonValueMapper` Jackson de código abierto. Puede usar este mapeador para procesar sus datos de Amazon Ion al realizar transacciones de lectura y escritura.

El archivo `CreateLedger.java` también depende del siguiente programa (`DescribeLedger.java`), que describe el estado actual de su libro mayor.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.model.DescribeLedgerRequest;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Describe a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class DescribeLedger {
    public static AmazonQLDB client = CreateLedger.getClient();
    public static final Logger log = LoggerFactory.getLogger(DescribeLedger.class);

    private DescribeLedger() { }

    public static void main(final String... args) {
        try {

            describe(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to describe a ledger!", e);
        }
    }
```

```
    /**
     * Describe a ledger.
     *
     * @param name
     *              Name of the ledger to describe.
     * @return {@link DescribeLedgerResult} from QLDB.
     */
    public static DescribeLedgerResult describe(final String name) {
        log.info("Let's describe ledger with name: {}...", name);
        DescribeLedgerRequest request = new DescribeLedgerRequest().withName(name);
        DescribeLedgerResult result = client.describeLedger(request);
        log.info("Success. Ledger description: {}", result);
        return result;
    }
}
```

2. Compile y ejecute el programa `CreateLedger.java` para crear un libro mayor denominado `vehicle-registration`.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.CreateLedgerRequest;
import com.amazonaws.services.qldb.model.CreateLedgerResult;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;
import com.amazonaws.services.qldb.model.LedgerState;
import com.amazonaws.services.qldb.model.PermissionsMode;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Create a ledger and wait for it to be active.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateLedger {
    public static final Logger log =
 LoggerFactory.getLogger(CreateLedger.class);
    public static final Long LEDGER_CREATION_POLL_PERIOD_MS = 10_000L;
    public static String endpoint = null;
    public static String region = null;
    public static AmazonQLDB client = getClient();

    private CreateLedger() {
    }

    /**
     * Build a low-level QLDB client.
     *
     * @return {@link AmazonQLDB} control plane client.
     */
    public static AmazonQLDB getClient() {
        AmazonQLDBClientBuilder builder = AmazonQLDBClientBuilder.standard();
        if (null != endpoint && null != region) {
```

```
            builder.setEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration(endpoint, region));
        }
        return builder.build();
    }

    public static void main(final String... args) throws Exception {
        try {
            client = getClient();

            create(Constants.LEDGER_NAME);

            waitForActive(Constants.LEDGER_NAME);

        } catch (Exception e) {
            log.error("Unable to create the ledger!", e);
            throw e;
        }
    }

    /**
     * Create a new ledger with the specified ledger name.
     *
     * @param ledgerName Name of the ledger to be created.
     * @return {@link CreateLedgerResult} from QLDB.
     */
    public static CreateLedgerResult create(final String ledgerName) {
        log.info("Let's create the ledger with name: {}...", ledgerName);
        CreateLedgerRequest request = new CreateLedgerRequest()
                .withName(ledgerName)
                .withPermissionsMode(PermissionsMode.ALLOW_ALL);
        CreateLedgerResult result = client.createLedger(request);
        log.info("Success. Ledger state: {}.", result.getState());
        return result;
    }

    /**
     * Wait for a newly created ledger to become active.
     *
     * @param ledgerName Name of the ledger to wait on.
     * @return {@link DescribeLedgerResult} from QLDB.
     * @throws InterruptedException if thread is being interrupted.
     */
```

```
    public static DescribeLedgerResult waitForActive(final String ledgerName)
 throws InterruptedException {
        log.info("Waiting for ledger to become active...");
        while (true) {
            DescribeLedgerResult result = DescribeLedger.describe(ledgerName);
            if (result.getState().equals(LedgerState.ACTIVE.name())) {
                log.info("Success. Ledger is active and ready to use.");
                return result;
            }
            log.info("The ledger is still creating. Please wait...");
            Thread.sleep(LEDGER_CREATION_POLL_PERIOD_MS);
        }
    }
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial;

import com.amazonaws.services.qldb.AmazonQLDB;
import com.amazonaws.services.qldb.AmazonQLDBClientBuilder;
import com.amazonaws.services.qldb.model.CreateLedgerRequest;
import com.amazonaws.services.qldb.model.CreateLedgerResult;
import com.amazonaws.services.qldb.model.DescribeLedgerResult;
import com.amazonaws.services.qldb.model.LedgerState;
import com.amazonaws.services.qldb.model.PermissionsMode;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Create a ledger and wait for it to be active.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateLedger {
    public static final Logger log =
 LoggerFactory.getLogger(CreateLedger.class);
    public static final Long LEDGER_CREATION_POLL_PERIOD_MS = 10_000L;
    public static AmazonQLDB client = getClient();

    private CreateLedger() { }

    /**
     * Build a low-level QLDB client.
     *
     * @return {@link AmazonQLDB} control plane client.
     */
    public static AmazonQLDB getClient() {
        return AmazonQLDBClientBuilder.standard().build();
    }

    public static void main(final String... args) throws Exception {
        try {

            create(Constants.LEDGER_NAME);

            waitForActive(Constants.LEDGER_NAME);
```

```
        } catch (Exception e) {
            log.error("Unable to create the ledger!", e);
            throw e;
        }
    }

    /**
     * Create a new ledger with the specified ledger name.
     *
     * @param ledgerName
     *              Name of the ledger to be created.
     * @return {@link CreateLedgerResult} from QLDB.
     */
    public static CreateLedgerResult create(final String ledgerName) {
        log.info("Let's create the ledger with name: {}...", ledgerName);
        CreateLedgerRequest request = new CreateLedgerRequest()
                .withName(ledgerName)
                .withPermissionsMode(PermissionsMode.ALLOW_ALL);
        CreateLedgerResult result = client.createLedger(request);
        log.info("Success. Ledger state: {}.", result.getState());
        return result;
    }

    /**
     * Wait for a newly created ledger to become active.
     *
     * @param ledgerName
     *              Name of the ledger to wait on.
     * @return {@link DescribeLedgerResult} from QLDB.
     * @throws InterruptedException if thread is being interrupted.
     */
    public static DescribeLedgerResult waitForActive(final String ledgerName)
throws InterruptedException {
        log.info("Waiting for ledger to become active...");
        while (true) {
            DescribeLedgerResult result = DescribeLedger.describe(ledgerName);
            if (result.getState().equals(LedgerState.ACTIVE.name())) {
                log.info("Success. Ledger is active and ready to use.");
                return result;
            }
            log.info("The ledger is still creating. Please wait...");
            Thread.sleep(LEDGER_CREATION_POLL_PERIOD_MS);
        }
    }
```

```
}
```

> **Note**
>
> - En la llamada `createLedger`, debe especificar un nombre de libro mayor y un modo de permisos. Recomendamos encarecidamente el modo de permisos `STANDARD` para maximizar la seguridad de los datos del libro mayor.
> - Al crear un libro mayor, se habilita de forma predeterminada la protección contra la eliminación. Se trata de una característica de QLDB que impide que los libros mayores sean eliminados por cualquier usuario. Tiene la opción de deshabilitar la protección contra la eliminación al crear el libro mayor con la API QLDB o AWS Command Line Interface (AWS CLI).
> - Si lo desea, también puede especificar etiquetas para adjuntar al libro mayor.

Para comprobar la conexión con el nuevo libro mayor, continúe con Paso 2: para probar la conectividad con el libro mayor.

Paso 2: para probar la conectividad con el libro mayor

En este paso, verifica que puede conectarse al libro mayor `vehicle-registration` de Amazon QLDB mediante el punto de conexión de la API de datos transaccionales.

Para probar la conectividad con el libro mayor

1. Utilice el siguiente programa (`ConnectToLedger.java`) para crear una conexión de sesión de datos con el libro mayor `vehicle-registration`.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
of this
 * software and associated documentation files (the "Software"), to deal in the
Software
```

```
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.net.URI;
import java.net.URISyntaxException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.services.qldbsession.QldbSessionClient;
import software.amazon.awssdk.services.qldbsession.QldbSessionClientBuilder;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.RetryPolicy;

/**
 * Connect to a session for a given ledger using default settings.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ConnectToLedger {
    public static final Logger log =
 LoggerFactory.getLogger(ConnectToLedger.class);
    public static AwsCredentialsProvider credentialsProvider;
    public static String endpoint = null;
    public static String ledgerName = Constants.LEDGER_NAME;
    public static String region = null;
```

```
    public static QldbDriver driver;

    private ConnectToLedger() {
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @param retryAttempts How many times the transaction will be retried in
     * case of a retryable issue happens like Optimistic Concurrency Control
exception,
     * server side failures or network issues.
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver(int retryAttempts) {
        QldbSessionClientBuilder builder = getAmazonQldbSessionClientBuilder();
        return QldbDriver.builder()
                             .ledger(ledgerName)
                             .transactionRetryPolicy(RetryPolicy
                                   .builder()
                                   .maxRetries(retryAttempts)
                                   .build())
                             .sessionClientBuilder(builder)
                             .build();
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver createQldbDriver() {
        QldbSessionClientBuilder builder = getAmazonQldbSessionClientBuilder();
        return QldbDriver.builder()
            .ledger(ledgerName)
            .transactionRetryPolicy(RetryPolicy.builder()

.maxRetries(Constants.RETRY_LIMIT).build())
            .sessionClientBuilder(builder)
            .build();
    }

    /**
```

```
     * Creates a QldbSession builder that is passed to the QldbDriver to connect
 to the Ledger.
     *
     * @return An instance of the AmazonQLDBSessionClientBuilder
     */
    public static QldbSessionClientBuilder getAmazonQldbSessionClientBuilder() {
        QldbSessionClientBuilder builder = QldbSessionClient.builder();
        if (null != endpoint && null != region) {
            try {
                builder.endpointOverride(new URI(endpoint));
            } catch (URISyntaxException e) {
                throw new IllegalArgumentException(e);
            }
        }
        if (null != credentialsProvider) {
            builder.credentialsProvider(credentialsProvider);
        }
        return builder;
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static QldbDriver getDriver() {
        if (driver == null) {
            driver = createQldbDriver();
        }
        return driver;
    }


    public static void main(final String... args) {
        Iterable<String> tables = ConnectToLedger.getDriver().getTableNames();
        log.info("Existing tables in the ledger:");
        for (String table : tables) {
            log.info("- {} ", table);
        }
    }
}
```

> **Note**
>
> - Para ejecutar operaciones de datos en su libro mayor, debe crear una instancia de la clase `QldbDriver` para conectarse a un libro mayor específico. Se trata de un objeto de cliente diferente al cliente de `AmazonQLDB` que utilizó en el paso anterior para crear el libro mayor. Ese cliente anterior solo se usa para ejecutar las operaciones de la API de administración que se enumeran en Referencia de la API de Amazon QLDB.
> - En primer lugar, debe crear un objeto `QldbDriver`. Debe especificar un nombre de libro mayor al crear este controlador.
>
>   A continuación, puede utilizar el método `execute` de este controlador para ejecutar instrucciones PartiQL.
> - Si lo desea, puede especificar un número máximo de reintentos para las excepciones de transacción. El método `execute` reintenta automáticamente los conflictos de control de concurrencia optimista (OCC) y otras excepciones transitorias frecuentes hasta este límite configurable. El valor predeterminado es `4`.
>
>   Si la transacción sigue fallando una vez alcanzado el límite, el controlador lanza la excepción. Para obtener más información, consulte Descripción de la política de reintentos del controlador en Amazon QLDB.

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
```

```
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.qldbsession.AmazonQLDBSessionClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.PooledQldbDriver;
import software.amazon.qldb.QldbDriver;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.exceptions.QldbClientException;

/**
 * Connect to a session for a given ledger using default settings.
 * <p>
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class ConnectToLedger {
    public static final Logger log =
 LoggerFactory.getLogger(ConnectToLedger.class);
    public static AWSCredentialsProvider credentialsProvider;
    public static String endpoint = null;
    public static String ledgerName = Constants.LEDGER_NAME;
    public static String region = null;
    private static PooledQldbDriver driver;

    private ConnectToLedger() {
    }
```

```
    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static PooledQldbDriver createQldbDriver() {
        AmazonQLDBSessionClientBuilder builder =
AmazonQLDBSessionClientBuilder.standard();
        if (null != endpoint && null != region) {
            builder.setEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration(endpoint, region));
        }
        if (null != credentialsProvider) {
            builder.setCredentials(credentialsProvider);
        }
        return PooledQldbDriver.builder()
                .withLedger(ledgerName)
                .withRetryLimit(Constants.RETRY_LIMIT)
                .withSessionClientBuilder(builder)
                .build();
    }

    /**
     * Create a pooled driver for creating sessions.
     *
     * @return The pooled driver for creating sessions.
     */
    public static PooledQldbDriver getDriver() {
        if (driver == null) {
            driver = createQldbDriver();
        }
        return driver;
    }

    /**
     * Connect to a ledger through a {@link QldbDriver}.
     *
     * @return {@link QldbSession}.
     */
    public static QldbSession createQldbSession() {
        return getDriver().getSession();
    }

    public static void main(final String... args) {
```

```
        try (QldbSession qldbSession = createQldbSession()) {
            log.info("Listing table names ");
            for (String tableName : qldbSession.getTableNames()) {
                log.info(tableName);
            }
        } catch (QldbClientException e) {
            log.error("Unable to create session.", e);
        }
    }
}
```

Note

- Para ejecutar operaciones de datos en su libro mayor, debe crear una instancia de la clase `PooledQldbDriver` o `QldbDriver` para conectarse a un libro mayor específico. Se trata de un objeto de cliente diferente al cliente de `AmazonQLDB` que utilizó en el paso anterior para crear el libro mayor. Ese cliente anterior solo se usa para ejecutar las operaciones de la API de administración que se enumeran en Referencia de la API de Amazon QLDB.

  Recomendamos usar `PooledQldbDriver` a menos que necesite implementar un grupo de sesiones personalizado con `QldbDriver`. El tamaño predeterminado del grupo para `PooledQldbDriver` es el número máximo de conexiones HTTP abiertas que permite el cliente de sesión.

- En primer lugar, debe crear un objeto `PooledQldbDriver`. Debe especificar un nombre de libro mayor al crear este controlador.

  A continuación, puede utilizar el método `execute` de este controlador para ejecutar instrucciones PartiQL. O bien, puede crear manualmente una sesión a partir de este objeto controlador agrupado y utilizar el método `execute` de la sesión. Una sesión representa una conexión única con el libro mayor.

- Si lo desea, puede especificar un número máximo de reintentos para las excepciones de transacción. El método `execute` reintenta automáticamente los conflictos de control de concurrencia optimista (OCC) y otras excepciones transitorias frecuentes hasta este límite configurable. El valor predeterminado es `4`.

Si la transacción sigue fallando una vez alcanzado el límite, el controlador lanza la excepción. Para obtener más información, consulte Descripción de la política de reintentos del controlador en Amazon QLDB.

2. Compila y ejecuta el programa `ConnectToLedger.java` para probar la conectividad de la sesión de datos con el libro mayor `vehicle-registration`.

Anulación de Región de AWS

La aplicación de ejemplo se conecta a QLDB en su Región de AWS predeterminada, que puede configurar como se describe en Configurar la región y las credenciales AWS predeterminadas del paso de requisito previo. Puede cambiar la región modificando las propiedades del constructor del cliente de sesión QLDB.

2.x

En el siguiente ejemplo de código se crea una instancia de objeto de `QldbSessionClientBuilder` nuevo.

```
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);
```

Puede usar el método `region` para ejecutar el código en QLDB en cualquier región donde se encuentre disponible. Para obtener una lista completa, consulte puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

1.x

En el siguiente ejemplo de código se crea una instancia de objeto de `AmazonQLDBSessionClientBuilder` nuevo.

```
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);
```

Puede usar el método `withRegion` para ejecutar el código en QLDB en cualquier región donde se encuentre disponible. Para obtener una lista completa, consulte puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

Para crear tablas en el libro mayor `vehicle-registration`, continúe con Paso 3: cree tablas, índices y datos de muestra.

Paso 3: cree tablas, índices y datos de muestra

Cuando su libro mayor de Amazon QLDB esté activo y acepte conexiones, podrá empezar a crear tablas con datos sobre los vehículos, sus propietarios y su información de registro. Tras crear las tablas y los índices, puede cargarlos con datos.

En este paso, creará cuatro tablas en el libro mayor `vehicle-registration`:

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

También se crean los siguientes índices.

Nombre de la tabla	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Al insertar datos de ejemplo, primero debe insertar los documentos en la tabla `Person`. A continuación, utilizará los `id` asignados por el sistema de cada documento `Person` para rellenar los campos correspondientes en los documentos `VehicleRegistration` y `DriversLicense` correspondientes.

> **Tip**
>
> Como práctica recomendada, utilice un `id` de documento asignado por el sistema como clave externa. Si bien puede definir campos que pretenden ser identificadores únicos (por ejemplo, el número de chasis [VIN] de un vehículo), el verdadero identificador único de un documento es su `id`. Este campo se incluye en los metadatos del documento, que puede consultar en la vista confirmada (la vista de una tabla definida por el sistema).
> Para obtener más información acerca de las vistas en QLDB, consulte Conceptos clave.
> Para obtener más información sobre metadatos, consulte Consulta de los metadatos del documento.

Para configurar los datos de muestra

1. Revise los siguientes archivos `.java`. Estas clases de modelos representan los documentos que ha almacenado en las tablas `vehicle-registration`. Se pueden serializar desde y hacia el formato Amazon Ion.

   > **Note**
   >
   > Los Documentos de Amazon QLDB se almacenan en formato Ion, que es un superconjunto de JSON. Por lo tanto, puede usar la biblioteca FasterXML Jackson para modelar los datos en JSON.

   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. Revise el siguiente archivo (`SampleData.java`), que representa los datos de ejemplo que inserta en las tablas `vehicle-registration`.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
```

```
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.model;

import com.amazon.ion.IonString;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import java.io.IOException;
import java.math.BigDecimal;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.ConnectToLedger;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.qldb.DmlResultDocument;
import software.amazon.qldb.tutorial.qldb.QldbRevision;
```

```
/**
 * Sample domain objects for use throughout this tutorial.
 */
public final class SampleData {
    public static final DateTimeFormatter DATE_TIME_FORMAT =
 DateTimeFormatter.ofPattern("yyyy-MM-dd");

    public static final List<VehicleRegistration> REGISTRATIONS =
 Collections.unmodifiableList(Arrays.asList(
            new VehicleRegistration("1N4AL11D75C109151", "LEWISR261LL", "WA",
 "Seattle",
                    BigDecimal.valueOf(90.25), convertToLocalDate("2017-08-21"),
 convertToLocalDate("2020-05-11"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("KM8SRDHF6EU074761", "CA762X", "WA", "Kent",
                    BigDecimal.valueOf(130.75),
 convertToLocalDate("2017-09-14"), convertToLocalDate("2020-06-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("3HGGK5G53FM761765", "CD820Z", "WA",
 "Everett",
                    BigDecimal.valueOf(442.30),
 convertToLocalDate("2011-03-17"), convertToLocalDate("2021-03-24"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1HVBBAANXWH544237", "LS477D", "WA",
 "Tacoma",
                    BigDecimal.valueOf(42.20), convertToLocalDate("2011-10-26"),
 convertToLocalDate("2023-09-25"),
                    new Owners(new Owner(null), Collections.emptyList())),
            new VehicleRegistration("1C4RJFAG0FC625797", "TH393F", "WA",
 "Olympia",
                    BigDecimal.valueOf(30.45), convertToLocalDate("2013-09-02"),
 convertToLocalDate("2024-03-19"),
                    new Owners(new Owner(null), Collections.emptyList()))
    ));

    public static final List<Vehicle> VEHICLES =
 Collections.unmodifiableList(Arrays.asList(
            new Vehicle("1N4AL11D75C109151", "Sedan", 2011,  "Audi", "A5",
 "Silver"),
            new Vehicle("KM8SRDHF6EU074761", "Sedan", 2015, "Tesla", "Model S",
 "Blue"),
            new Vehicle("3HGGK5G53FM761765", "Motorcycle", 2011, "Ducati",
 "Monster 1200", "Yellow"),
```

```
            new Vehicle("1HVBBAANXWH544237", "Semi", 2009, "Ford", "F 150",
"Black"),
            new Vehicle("1C4RJFAG0FC625797", "Sedan", 2019, "Mercedes", "CLK
350", "White")
    ));

    public static final List<Person> PEOPLE =
Collections.unmodifiableList(Arrays.asList(
            new Person("Raul", "Lewis", convertToLocalDate("1963-08-19"),
                    "LEWISR261LL", "Driver License",  "1719 University Street,
Seattle, WA, 98109"),
            new Person("Brent", "Logan", convertToLocalDate("1967-07-03"),
                    "LOGANB486CG", "Driver License", "43 Stockert Hollow Road,
Everett, WA, 98203"),
            new Person("Alexis", "Pena", convertToLocalDate("1974-02-10"),
                    "744 849 301", "SSN", "4058 Melrose Street, Spokane Valley,
WA, 99206"),
            new Person("Melvin", "Parker", convertToLocalDate("1976-05-22"),
                    "P626-168-229-765", "Passport", "4362 Ryder Avenue, Seattle,
WA, 98101"),
            new Person("Salvatore", "Spencer", convertToLocalDate("1997-11-15"),
                    "S152-780-97-415-0", "Passport", "4450 Honeysuckle Lane,
Seattle, WA, 98101")
    ));

    public static final List<DriversLicense> LICENSES =
Collections.unmodifiableList(Arrays.asList(
            new DriversLicense(null, "LEWISR261LL", "Learner",
                    convertToLocalDate("2016-12-20"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "LOGANB486CG", "Probationary",
                    convertToLocalDate("2016-04-06"),
convertToLocalDate("2020-11-15")),
            new DriversLicense(null, "744 849 301", "Full",
                    convertToLocalDate("2017-12-06"),
convertToLocalDate("2022-10-15")),
            new DriversLicense(null, "P626-168-229-765", "Learner",
                    convertToLocalDate("2017-08-16"),
convertToLocalDate("2021-11-15")),
            new DriversLicense(null, "S152-780-97-415-0", "Probationary",
                    convertToLocalDate("2015-08-15"),
convertToLocalDate("2021-08-21"))
    ));
```

```
    private SampleData() { }

    /**
     * Converts a date string with the format 'yyyy-MM-dd' into a {@link
java.util.Date} object.
     *
     * @param date
     *              The date string to convert.
     * @return {@link java.time.LocalDate} or null if there is a {@link
ParseException}
     */
    public static synchronized LocalDate convertToLocalDate(String date) {
        return LocalDate.parse(date, DATE_TIME_FORMAT);
    }

    /**
     * Convert the result set into a list of IonValues.
     *
     * @param result
     *              The result set to convert.
     * @return a list of IonValues.
     */
    public static List<IonValue> toIonValues(Result result) {
        final List<IonValue> valueList = new ArrayList<>();
        result.iterator().forEachRemaining(valueList::add);
        return valueList;
    }

    /**
     * Get the document ID of a particular document.
     *
     * @param txn
     *              A transaction executor object.
     * @param tableName
     *              Name of the table containing the document.
     * @param identifier
     *              The identifier used to narrow down the search.
     * @param value
     *              Value of the identifier.
     * @return the list of document IDs in the result set.
     */
    public static String getDocumentId(final TransactionExecutor txn, final
String tableName,
```

```
                                        final String identifier, final String
value) {
       try {
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(value));
           final String query = String.format("SELECT metadata.id FROM
_ql_committed_%s AS p WHERE p.data.%s = ?",
                   tableName, identifier);
           Result result = txn.execute(query, parameters);
           if (result.isEmpty()) {
               throw new IllegalStateException("Unable to retrieve document ID
using " + value);
           }
           return getStringValueOfStructField((IonStruct)
result.iterator().next(), "id");
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Get the document by ID.
    *
    * @param tableName
    *              Name of the table to insert documents into.
    * @param documentId
    *              The unique ID of a document in the Person table.
    * @return a {@link QldbRevision} object.
    * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
    */
   public static QldbRevision getDocumentById(String tableName, String
documentId) {
       try {
           final IonValue ionValue =
Constants.MAPPER.writeValueAsIonValue(documentId);
           Result result = ConnectToLedger.getDriver().execute(txn -> {
               return txn.execute("SELECT c.* FROM _ql_committed_" + tableName
+ " AS c BY docId "
                                   + "WHERE docId = ?", ionValue);
           });
           if (result.isEmpty()) {
               throw new IllegalStateException("Unable to retrieve document by
id " + documentId + " in table " + tableName);
```

```
            }
            return Constants.MAPPER.readValue(result.iterator().next(),
QldbRevision.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Return a list of modified document IDs as strings from a DML {@link
Result}.
     *
     * @param result
     *              The result set from a DML operation.
     * @return the list of document IDs modified by the operation.
     */
    public static List<String> getDocumentIdsFromDmlResult(final Result result)
{
        final List<String> strings = new ArrayList<>();
        result.iterator().forEachRemaining(row ->
strings.add(getDocumentIdFromDmlResultDocument(row)));
        return strings;
    }

    /**
     * Convert the given DML result row's document ID to string.
     *
     * @param dmlResultDocument
     *              The {@link IonValue} representing the results of a DML
operation.
     * @return a string of document ID.
     */
    public static String getDocumentIdFromDmlResultDocument(final IonValue
dmlResultDocument) {
        try {
            DmlResultDocument result =
Constants.MAPPER.readValue(dmlResultDocument, DmlResultDocument.class);
            return result.getDocumentId();
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
```

```
    * Get the String value of a given {@link IonStruct} field name.
    * @param struct the {@link IonStruct} from which to get the value.
    * @param fieldName the name of the field from which to get the value.
    * @return the String value of the field within the given {@link IonStruct}.
    */
   public static String getStringValueOfStructField(final IonStruct struct,
final String fieldName) {
       return ((IonString) struct.get(fieldName)).stringValue();
   }

   /**
    * Return a copy of the given driver's license with updated person Id.
    *
    * @param oldLicense
    *              The old driver's license to update.
    * @param personId
    *              The PersonId of the driver.
    * @return the updated {@link DriversLicense}.
    */
   public static DriversLicense updatePersonIdDriversLicense(final
DriversLicense oldLicense, final String personId) {
       return new DriversLicense(personId, oldLicense.getLicenseNumber(),
oldLicense.getLicenseType(),
               oldLicense.getValidFromDate(), oldLicense.getValidToDate());
   }

   /**
    * Return a copy of the given vehicle registration with updated person Id.
    *
    * @param oldRegistration
    *              The old vehicle registration to update.
    * @param personId
    *              The PersonId of the driver.
    * @return the updated {@link VehicleRegistration}.
    */
   public static VehicleRegistration updateOwnerVehicleRegistration(final
VehicleRegistration oldRegistration,
                                                                    final
String personId) {
       return new VehicleRegistration(oldRegistration.getVin(),
oldRegistration.getLicensePlateNumber(),
               oldRegistration.getState(), oldRegistration.getCity(),
oldRegistration.getPendingPenaltyTicketAmount(),
```

```
                oldRegistration.getValidFromDate(),
 oldRegistration.getValidToDate(),
                new Owners(new Owner(personId), Collections.emptyList()));
    }
}
```

1.x

> **Important**
>
> Para el paquete Amazon Ion, debe usar el espacio de nombres `com.amazon.ion` de su aplicación. AWS SDK for Java depende de otro paquete de Ion en el espacio de nombres `software.amazon.ion`, pero se trata de un paquete heredado que no es compatible con el controlador QLDB.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> - Esta clase utiliza las bibliotecas de Ion para proporcionar métodos auxiliares que convierten los datos al formato Ion y desde él.
> - El método `getDocumentId` ejecuta una consulta en una tabla con el prefijo `_ql_committed_`. Se trata de un prefijo reservado que indica que desea consultar la vista confirmada de la tabla. En esta vista, los datos están anidados en el campo `data` y los metadatos están anidados en el campo `metadata`.

3. Compile y ejecute el siguiente programa (`CreateTable.java`) para crear las tablas mencionadas anteriormente.

2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create tables in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
```

```
 */
public final class CreateTable {
    public static final Logger log = LoggerFactory.getLogger(CreateTable.class);

    private CreateTable() { }

    /**
     * Registrations, vehicles, owners, and licenses tables being created in a
 single transaction.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @return the number of tables created.
     */
    public static int createTable(final TransactionExecutor txn, final String
 tableName) {
        log.info("Creating the '{}' table...", tableName);
        final String createTable = String.format("CREATE TABLE %s", tableName);
        final Result result = txn.execute(createTable);
        log.info("{} table created successfully.", tableName);
        return SampleData.toIonValues(result).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createTable(txn, Constants.DRIVERS_LICENSE_TABLE_NAME);
            createTable(txn, Constants.PERSON_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        });
    }
}
```

1.x

```
/*
 * Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create tables in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateTable {
    public static final Logger log = LoggerFactory.getLogger(CreateTable.class);

    private CreateTable() { }

    /**
     * Registrations, vehicles, owners, and licenses tables being created in a
 single transaction.
```

```
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @return the number of tables created.
     */
    public static int createTable(final TransactionExecutor txn, final String
 tableName) {
        log.info("Creating the '{}' table...", tableName);
        final String createTable = String.format("CREATE TABLE %s", tableName);
        final Result result = txn.execute(createTable);
        log.info("{} table created successfully.", tableName);
        return SampleData.toIonValues(result).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createTable(txn, Constants.DRIVERS_LICENSE_TABLE_NAME);
            createTable(txn, Constants.PERSON_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_TABLE_NAME);
            createTable(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
    }
}
```

> **Note**
>
> Este programa muestra cómo pasar una lambda `TransactionExecutor` al método `execute`. En este ejemplo, ejecuta varias instrucciones `CREATE TABLE` de PartiQL en una única transacción con una expresión lambda.
> El método `execute` inicia implícitamente una transacción, ejecuta todas las instrucciones en lambda y, a continuación, confirma automáticamente la transacción.

4. Compile y ejecute el siguiente programa (`CreateIndex.java`) para crear índices en las tablas, tal y como se describió anteriormente.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create indexes on tables in a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class CreateIndex {
    public static final Logger log = LoggerFactory.getLogger(CreateIndex.class);
```

```
    private CreateIndex() { }

    /**
     * In this example, create indexes for registrations and vehicles tables.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @param indexAttribute
     *              The index attribute to use.
     * @return the number of tables created.
     */
    public static int createIndex(final TransactionExecutor txn, final String
 tableName, final String indexAttribute) {
        log.info("Creating an index on {}...", indexAttribute);
        final String createIndex = String.format("CREATE INDEX ON %s (%s)",
 tableName, indexAttribute);
        final Result r = txn.execute(createIndex);
        return SampleData.toIonValues(r).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createIndex(txn, Constants.PERSON_TABLE_NAME,
 Constants.PERSON_GOV_ID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_NUMBER_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_PERSONID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME);
        });
        log.info("Indexes created successfully!");
    }
}
```

1.x

```
/*
 * Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Create indexes on tables in a particular ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
```

```
 */
public final class CreateIndex {
    public static final Logger log = LoggerFactory.getLogger(CreateIndex.class);

    private CreateIndex() { }

    /**
     * In this example, create indexes for registrations and vehicles tables.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to be created.
     * @param indexAttribute
     *              The index attribute to use.
     * @return the number of tables created.
     */
    public static int createIndex(final TransactionExecutor txn, final String
 tableName, final String indexAttribute) {
        log.info("Creating an index on {}...", indexAttribute);
        final String createIndex = String.format("CREATE INDEX ON %s (%s)",
 tableName, indexAttribute);
        final Result r = txn.execute(createIndex);
        return SampleData.toIonValues(r).size();
    }

    public static void main(final String... args) {
        ConnectToLedger.getDriver().execute(txn -> {
            createIndex(txn, Constants.PERSON_TABLE_NAME,
 Constants.PERSON_GOV_ID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_NUMBER_INDEX_NAME);
            createIndex(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.DRIVER_LICENSE_PERSONID_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VIN_INDEX_NAME);
            createIndex(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.VEHICLE_REGISTRATION_LICENSE_PLATE_NUMBER_INDEX_NAME);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Indexes created successfully!");
    }
```

```
}
```

5. Compile y ejecute el siguiente programa (`InsertDocument.java`) para insertar los datos de ejemplo en las tablas.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import com.amazon.ion.IonValue;

import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.DriversLicense;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

/**
 * Insert documents into a table in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class InsertDocument {
    public static final Logger log =
 LoggerFactory.getLogger(InsertDocument.class);

    private InsertDocument() { }

    /**
     * Insert the given list of documents into the specified table and return
 the document IDs of the inserted documents.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documents
     *              List of documents to insert into the specified table.
     * @return a list of document IDs.
     * @throws IllegalStateException if failed to convert documents into an
 {@link IonValue}.
     */
    public static List<String> insertDocuments(final TransactionExecutor txn,
 final String tableName,
                                               final List documents) {
        log.info("Inserting some documents in the {} table...", tableName);
        try {
            final String query = String.format("INSERT INTO %s ?", tableName);
            final IonValue ionDocuments =
 Constants.MAPPER.writeValueAsIonValue(documents);
```

```
            return SampleData.getDocumentIdsFromDmlResult(txn.execute(query,
ionDocuments));
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update PersonIds in driver's licenses and in vehicle registrations using
document IDs.
     *
     * @param documentIds
     *              List of document IDs representing the PersonIds in
DriversLicense and PrimaryOwners in VehicleRegistration.
     * @param licenses
     *              List of driver's licenses to update.
     * @param registrations
     *              List of registrations to update.
     */
    public static void updatePersonId(final List<String> documentIds, final
List<DriversLicense> licenses,
                                      final List<VehicleRegistration>
registrations) {
        for (int i = 0; i < documentIds.size(); ++i) {
            DriversLicense license = SampleData.LICENSES.get(i);
            VehicleRegistration registration = SampleData.REGISTRATIONS.get(i);
            licenses.add(SampleData.updatePersonIdDriversLicense(license,
documentIds.get(i)));

registrations.add(SampleData.updateOwnerVehicleRegistration(registration,
documentIds.get(i)));
        }
    }

    public static void main(final String... args) {
        final List<DriversLicense> newDriversLicenses = new ArrayList<>();
        final List<VehicleRegistration> newVehicleRegistrations = new
ArrayList<>();
        ConnectToLedger.getDriver().execute(txn -> {
            List<String> documentIds = insertDocuments(txn,
Constants.PERSON_TABLE_NAME, SampleData.PEOPLE);
            updatePersonId(documentIds, newDriversLicenses,
newVehicleRegistrations);
```

```
            insertDocuments(txn, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLES);
            insertDocuments(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                    Collections.unmodifiableList(newVehicleRegistrations));
            insertDocuments(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
                    Collections.unmodifiableList(newDriversLicenses));
        });
        log.info("Documents inserted successfully!");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonValue;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.qldb.QldbSession;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.DriversLicense;
import software.amazon.qldb.tutorial.model.SampleData;
import software.amazon.qldb.tutorial.model.VehicleRegistration;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Insert documents into a table in a QLDB ledger.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class InsertDocument {
    public static final Logger log =
 LoggerFactory.getLogger(InsertDocument.class);

    private InsertDocument() { }

    /**
     * Insert the given list of documents into the specified table and return
 the document IDs of the inserted documents.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param tableName
     *              Name of the table to insert documents into.
     * @param documents
     *              List of documents to insert into the specified table.
     * @return a list of document IDs.
     * @throws IllegalStateException if failed to convert documents into an
 {@link IonValue}.
     */
    public static List<String> insertDocuments(final TransactionExecutor txn,
 final String tableName,
                                               final List documents) {
        log.info("Inserting some documents in the {} table...", tableName);
        try {
```

```
            final String statement = String.format("INSERT INTO %s ?",
tableName);
            final IonValue ionDocuments =
Constants.MAPPER.writeValueAsIonValue(documents);
            final List<IonValue> parameters =
Collections.singletonList(ionDocuments);
            return SampleData.getDocumentIdsFromDmlResult(txn.execute(statement,
parameters));
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update PersonIds in driver's licenses and in vehicle registrations using
document IDs.
     *
     * @param documentIds
     *              List of document IDs representing the PersonIds in
DriversLicense and PrimaryOwners in VehicleRegistration.
     * @param licenses
     *              List of driver's licenses to update.
     * @param registrations
     *              List of registrations to update.
     */
    public static void updatePersonId(final List<String> documentIds, final
List<DriversLicense> licenses,
                                      final List<VehicleRegistration>
registrations) {
        for (int i = 0; i < documentIds.size(); ++i) {
            DriversLicense license = SampleData.LICENSES.get(i);
            VehicleRegistration registration = SampleData.REGISTRATIONS.get(i);
            licenses.add(SampleData.updatePersonIdDriversLicense(license,
documentIds.get(i)));

registrations.add(SampleData.updateOwnerVehicleRegistration(registration,
documentIds.get(i)));
        }
    }

    public static void main(final String... args) {
        final List<DriversLicense> newDriversLicenses = new ArrayList<>();
        final List<VehicleRegistration> newVehicleRegistrations = new
ArrayList<>();
```

```
        ConnectToLedger.getDriver().execute(txn -> {
            List<String> documentIds = insertDocuments(txn,
 Constants.PERSON_TABLE_NAME, SampleData.PEOPLE);
            updatePersonId(documentIds, newDriversLicenses,
 newVehicleRegistrations);
            insertDocuments(txn, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLES);
            insertDocuments(txn, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
                    Collections.unmodifiableList(newVehicleRegistrations));
            insertDocuments(txn, Constants.DRIVERS_LICENSE_TABLE_NAME,
                    Collections.unmodifiableList(newDriversLicenses));
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Documents inserted successfully!");
    }
}
```

> **Note**
>
> - Este programa muestra cómo llamar a la función `execute` con valores parametrizados. Puede pasar parámetros de datos de tipo `IonValue` además de la instrucción PartiQL que desee ejecutar. Utilice un signo de interrogación (`?`) como marcador de posición variable en la cadena de la instrucción.
> - Si una instrucción `INSERT` tiene éxito, devuelve el `id` de cada documento insertado.

A continuación, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`. Continúe en Paso 4: consultar las tablas en un libro mayor.

Paso 4: consultar las tablas en un libro mayor

Tras crear tablas en un libro mayor de Amazon QLDB y cargarlas con datos, puede ejecutar consultas para revisar los datos de registro del vehículo que acaba de insertar. QLDB emplea PartiQL como lenguaje de consulta y Amazon Ion como modelo de datos orientado a documentos.

PartiQL es un lenguaje de consulta de código abierto compatible con SQL que se ha ampliado para funcionar con Ion. PartiQL le permite insertar, consultar y administrar sus datos con operadores SQL conocidos. Amazon Ion es un superconjunto de JSON. Ion es un formato de datos de código abierto basado en documentos que le brinda la flexibilidad de almacenar y procesar datos estructurados, semiestructurados y anidados.

En este paso, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`.

> **Warning**
>
> Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. PartiQL admite este tipo de consultas porque es compatible con SQL. Sin embargo, no ejecute escaneados de tablas para casos de uso de producción en QLDB. Los escaneos de tablas pueden provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad en un campo indexado o en un ID de documento, por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Para consultar las tablas

- Compile y ejecute el siguiente programa (`FindVehicles.java`) para consultar todos los vehículos registrados a nombre de una persona en su libro mayor.

  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**
>
> En primer lugar, este programa consulta la tabla `Person` del documento con `GovId` `LEWISR261LL` para obtener su campo de metadatos `id`.
> A continuación, utiliza este documento `id` como clave externa para consultar la tabla `VehicleRegistration` mediante `PrimaryOwner.PersonId`. También combina `VehicleRegistration` con la tabla `Vehicle` por el campo `VIN`.

Para obtener información sobre la modificación de los documentos en las tablas del libro mayor `vehicle-registration`, consulte Paso 5: modificar los documentos de un libro mayor.

Paso 5: modificar los documentos de un libro mayor

Ahora que tiene datos con los que trabajar, puede empezar a realizar cambios en los documentos del libro mayor `vehicle-registration` de Amazon QLDB. En este paso, los siguientes ejemplos de código muestran cómo ejecutar instrucciones de lenguaje de manipulación de datos (DML). Estas instrucciones actualizan al propietario principal de un vehículo y añaden un propietario secundario a otro vehículo.

Para modificar documentos

1. Compile y ejecute el siguiente programa (`TransferVehicleOwnership.java`) para actualizar el VIN del propietario principal del vehículo `1N4AL11D75C109151` en su libro mayor.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonReaderBuilder;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class TransferVehicleOwnership {
    public static final Logger log =
 LoggerFactory.getLogger(TransferVehicleOwnership.class);

    private TransferVehicleOwnership() { }

    /**
     * Query a driver's information using the given ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param documentId
```

```
     *              The unique ID of a document in the Person table.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static Person findPersonFromDocumentId(final TransactionExecutor txn,
final String documentId) {
        try {
            log.info("Finding person for documentId: {}...", documentId);
            final String query = "SELECT p.* FROM Person AS p BY pid WHERE pid
= ?";

            Result result = txn.execute(query,
Constants.MAPPER.writeValueAsIonValue(documentId));
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to find person with ID:
" + documentId);
            }

            return Constants.MAPPER.readValue(result.iterator().next(),
Person.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Find the primary owner for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static Person findPrimaryOwnerForVehicle(final TransactionExecutor
txn, final String vin) {
        try {
            log.info("Finding primary owner for vehicle with Vin: {}...", vin);
            final String query = "SELECT Owners.PrimaryOwner.PersonId FROM
VehicleRegistration AS v WHERE v.VIN = ?";
```

```
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            Result result = txn.execute(query, parameters);
            final List<IonStruct> documents = ScanTable.toIonStructs(result);
            ScanTable.printDocuments(documents);
            if (documents.isEmpty()) {
                throw new IllegalStateException("Unable to find registrations
with VIN: " + vin);
            }

            final IonReader reader =
IonReaderBuilder.standard().build(documents.get(0));
            final String personId = Constants.MAPPER.readValue(reader,
LinkedHashMap.class).get("PersonId").toString();
            return findPersonFromDocumentId(txn, personId);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update the primary owner for a vehicle registration with the given
documentId.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param documentId
     *              New PersonId for the primary owner.
     * @throws IllegalStateException if no vehicle registration was found using
the given document ID and VIN, or if failed
     * to convert parameters into {@link IonValue}.
     */
    public static void updateVehicleRegistration(final TransactionExecutor txn,
final String vin, final String documentId) {
        try {
            log.info("Updating primary owner for vehicle with Vin: {}...", vin);
            final String query = "UPDATE VehicleRegistration AS v SET
v.Owners.PrimaryOwner = ? WHERE v.VIN = ?";

            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(Constants.MAPPER.writeValueAsIonValue(new
Owner(documentId)));
```

```
            parameters.add(Constants.MAPPER.writeValueAsIonValue(vin));

            Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to transfer vehicle,
 could not find registration.");
            } else {
                log.info("Successfully transferred vehicle with VIN '{}' to new
 owner.", vin);
            }
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(0).getVin();
        final String primaryOwnerGovId = SampleData.PEOPLE.get(0).getGovId();
        final String newPrimaryOwnerGovId = SampleData.PEOPLE.get(1).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final Person primaryOwner = findPrimaryOwnerForVehicle(txn, vin);
            if (!primaryOwner.getGovId().equals(primaryOwnerGovId)) {
                // Verify the primary owner.
                throw new IllegalStateException("Incorrect primary owner
 identified for vehicle, unable to transfer.");
            }

            final String newOwner = Person.getDocumentIdByGovId(txn,
 newPrimaryOwnerGovId);
            updateVehicleRegistration(txn, vin, newOwner);
        });
        log.info("Successfully transferred vehicle ownership!");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
```

```
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import com.amazon.ion.IonReader;
import com.amazon.ion.IonStruct;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonReaderBuilder;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;
```

```
/**
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class TransferVehicleOwnership {
    public static final Logger log =
 LoggerFactory.getLogger(TransferVehicleOwnership.class);

    private TransferVehicleOwnership() { }

    /**
     * Query a driver's information using the given ID.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param documentId
     *              The unique ID of a document in the Person table.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
 IonValue}.
     */
    public static Person findPersonFromDocumentId(final TransactionExecutor txn,
 final String documentId) {
        try {
            log.info("Finding person for documentId: {}...", documentId);
            final String query = "SELECT p.* FROM Person AS p BY pid WHERE pid
 = ?";

            Result result = txn.execute(query,
 Constants.MAPPER.writeValueAsIonValue(documentId));
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to find person with ID:
 " + documentId);
            }

            return Constants.MAPPER.readValue(result.iterator().next(),
 Person.class);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
```

```
    }

    /**
     * Find the primary owner for the given VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @return a {@link Person} object.
     * @throws IllegalStateException if failed to convert parameter into {@link
IonValue}.
     */
    public static Person findPrimaryOwnerForVehicle(final TransactionExecutor
txn, final String vin) {
        try {
            log.info("Finding primary owner for vehicle with Vin: {}...", vin);
            final String query = "SELECT Owners.PrimaryOwner.PersonId FROM
VehicleRegistration AS v WHERE v.VIN = ?";
            final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            Result result = txn.execute(query, parameters);
            final List<IonStruct> documents = ScanTable.toIonStructs(result);
            ScanTable.printDocuments(documents);
            if (documents.isEmpty()) {
                throw new IllegalStateException("Unable to find registrations
with VIN: " + vin);
            }

            final IonReader reader =
IonReaderBuilder.standard().build(documents.get(0));
            final String personId = Constants.MAPPER.readValue(reader,
LinkedHashMap.class).get("PersonId").toString();
            return findPersonFromDocumentId(txn, personId);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Update the primary owner for a vehicle registration with the given
documentId.
     *
     * @param txn
```

```
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param documentId
     *              New PersonId for the primary owner.
     * @throws IllegalStateException if no vehicle registration was found using
the given document ID and VIN, or if failed
     * to convert parameters into {@link IonValue}.
     */
    public static void updateVehicleRegistration(final TransactionExecutor txn,
final String vin, final String documentId) {
        try {
            log.info("Updating primary owner for vehicle with Vin: {}...", vin);
            final String query = "UPDATE VehicleRegistration AS v SET
v.Owners.PrimaryOwner = ? WHERE v.VIN = ?";

            final List<IonValue> parameters = new ArrayList<>();
            parameters.add(Constants.MAPPER.writeValueAsIonValue(new
Owner(documentId)));
            parameters.add(Constants.MAPPER.writeValueAsIonValue(vin));

            Result result = txn.execute(query, parameters);
            ScanTable.printDocuments(result);
            if (result.isEmpty()) {
                throw new IllegalStateException("Unable to transfer vehicle,
could not find registration.");
            } else {
                log.info("Successfully transferred vehicle with VIN '{}' to new
owner.", vin);
            }
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(0).getVin();
        final String primaryOwnerGovId = SampleData.PEOPLE.get(0).getGovId();
        final String newPrimaryOwnerGovId = SampleData.PEOPLE.get(1).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final Person primaryOwner = findPrimaryOwnerForVehicle(txn, vin);
            if (!primaryOwner.getGovId().equals(primaryOwnerGovId)) {
                // Verify the primary owner.
```

```
                throw new IllegalStateException("Incorrect primary owner
 identified for vehicle, unable to transfer.");
            }

            final String newOwner = Person.getDocumentIdByGovId(txn,
 newPrimaryOwnerGovId);
            updateVehicleRegistration(txn, vin, newOwner);
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Successfully transferred vehicle ownership!");
    }
}
```

2. Compile y ejecute el siguiente programa (`AddSecondaryOwner.java`) para añadir un propietario secundario al vehículo con el VIN `KM8SRDHF6EU074761` en su libro mayor.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
import software.amazon.qldb.tutorial.model.Owners;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Finds and adds secondary owners for a vehicle.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class AddSecondaryOwner {
    public static final Logger log =
 LoggerFactory.getLogger(AddSecondaryOwner.class);

    private AddSecondaryOwner() { }

    /**
     * Check whether a secondary owner has already been registered for the given
 VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwnerId
     *              The secondary owner to add.
     * @return {@code true} if the given secondary owner has already been
 registered, {@code false} otherwise.
```

```
    * @throws IllegalStateException if failed to convert VIN to an {@link
IonValue}.
    */
   public static boolean isSecondaryOwnerForVehicle(final TransactionExecutor
txn, final String vin,
                                                    final String
secondaryOwnerId) {
       try {
           log.info("Finding secondary owners for vehicle with VIN: {}...",
vin);
           final String query = "SELECT Owners.SecondaryOwners FROM
VehicleRegistration AS v WHERE v.VIN = ?";
           final List<IonValue> parameters =
Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
           final Result result = txn.execute(query, parameters);
           final Iterator<IonValue> itr = result.iterator();
           if (!itr.hasNext()) {
               return false;
           }

           final Owners owners = Constants.MAPPER.readValue(itr.next(),
Owners.class);
           if (null != owners.getSecondaryOwners()) {
               for (Owner owner : owners.getSecondaryOwners()) {
                   if (secondaryOwnerId.equalsIgnoreCase(owner.getPersonId()))
{
                       return true;
                   }
               }
           }

           return false;
       } catch (IOException ioe) {
           throw new IllegalStateException(ioe);
       }
   }

   /**
    * Adds a secondary owner for the specified VIN.
    *
    * @param txn
    *              The {@link TransactionExecutor} for lambda execute.
    * @param vin
    *              Unique VIN for a vehicle.
```

```
     * @param secondaryOwner
     *              The secondary owner to add.
     * @throws IllegalStateException if failed to convert parameter into an
 {@link IonValue}.
     */
    public static void addSecondaryOwnerForVin(final TransactionExecutor txn,
 final String vin,
                                               final String secondaryOwner) {
        try {
            log.info("Inserting secondary owner for vehicle with VIN: {}...",
 vin);
            final String query = String.format("FROM VehicleRegistration AS v
 WHERE v.VIN = ?" +
                    "INSERT INTO v.Owners.SecondaryOwners VALUE ?");
            final IonValue newOwner = Constants.MAPPER.writeValueAsIonValue(new
 Owner(secondaryOwner));
            final IonValue vinAsIonValue =
 Constants.MAPPER.writeValueAsIonValue(vin);
            Result result = txn.execute(query, vinAsIonValue, newOwner);
            log.info("VehicleRegistration Document IDs which had secondary
 owners added: ");
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(1).getVin();
        final String govId = SampleData.PEOPLE.get(0).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            if (isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log.info("Person with ID {} has already been added as a
 secondary owner of this vehicle.", govId);
            } else {
                addSecondaryOwnerForVin(txn, vin, documentId);
            }
        });
        log.info("Secondary owners successfully updated.");
    }
}
```

1.x

```
/*
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial;

import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.amazon.ion.IonValue;

import software.amazon.qldb.Result;
import software.amazon.qldb.TransactionExecutor;
import software.amazon.qldb.tutorial.model.Owner;
```

```
import software.amazon.qldb.tutorial.model.Owners;
import software.amazon.qldb.tutorial.model.Person;
import software.amazon.qldb.tutorial.model.SampleData;

/**
 * Finds and adds secondary owners for a vehicle.
 *
 * This code expects that you have AWS credentials setup per:
 * http://docs.aws.amazon.com/java-sdk/latest/developer-guide/setup-
credentials.html
 */
public final class AddSecondaryOwner {
    public static final Logger log =
 LoggerFactory.getLogger(AddSecondaryOwner.class);

    private AddSecondaryOwner() { }

    /**
     * Check whether a secondary owner has already been registered for the given
 VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwnerId
     *              The secondary owner to add.
     * @return {@code true} if the given secondary owner has already been
 registered, {@code false} otherwise.
     * @throws IllegalStateException if failed to convert VIN to an {@link
 IonValue}.
     */
    public static boolean isSecondaryOwnerForVehicle(final TransactionExecutor
 txn, final String vin,
                                                     final String
 secondaryOwnerId) {
        try {
            log.info("Finding secondary owners for vehicle with VIN: {}...",
 vin);
            final String query = "SELECT Owners.SecondaryOwners FROM
 VehicleRegistration AS v WHERE v.VIN = ?";
            final List<IonValue> parameters =
 Collections.singletonList(Constants.MAPPER.writeValueAsIonValue(vin));
            final Result result = txn.execute(query, parameters);
```

```
            final Iterator<IonValue> itr = result.iterator();
            if (!itr.hasNext()) {
                return false;
            }

            final Owners owners = Constants.MAPPER.readValue(itr.next(),
Owners.class);
            if (null != owners.getSecondaryOwners()) {
                for (Owner owner : owners.getSecondaryOwners()) {
                    if (secondaryOwnerId.equalsIgnoreCase(owner.getPersonId()))
{
                        return true;
                    }
                }
            }

            return false;
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    /**
     * Adds a secondary owner for the specified VIN.
     *
     * @param txn
     *              The {@link TransactionExecutor} for lambda execute.
     * @param vin
     *              Unique VIN for a vehicle.
     * @param secondaryOwner
     *              The secondary owner to add.
     * @throws IllegalStateException if failed to convert parameter into an
{@link IonValue}.
     */
    public static void addSecondaryOwnerForVin(final TransactionExecutor txn,
final String vin,
                                               final String secondaryOwner) {
        try {
            log.info("Inserting secondary owner for vehicle with VIN: {}...",
vin);
            final String query = String.format("FROM VehicleRegistration AS v
WHERE v.VIN = '%s' " +
                    "INSERT INTO v.Owners.SecondaryOwners VALUE ?", vin);
```

```
            final IonValue newOwner = Constants.MAPPER.writeValueAsIonValue(new
 Owner(secondaryOwner));
            Result result = txn.execute(query, newOwner);
            log.info("VehicleRegistration Document IDs which had secondary
 owners added: ");
            ScanTable.printDocuments(result);
        } catch (IOException ioe) {
            throw new IllegalStateException(ioe);
        }
    }

    public static void main(final String... args) {
        final String vin = SampleData.VEHICLES.get(1).getVin();
        final String govId = SampleData.PEOPLE.get(0).getGovId();

        ConnectToLedger.getDriver().execute(txn -> {
            final String documentId = Person.getDocumentIdByGovId(txn, govId);
            if (isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log.info("Person with ID {} has already been added as a
 secondary owner of this vehicle.", govId);
            } else {
                addSecondaryOwnerForVin(txn, vin, documentId);
            }
        }, (retryAttempt) -> log.info("Retrying due to OCC conflict..."));
        log.info("Secondary owners successfully updated.");
    }
}
```

Para revisar estos cambios en el libro mayor `vehicle-registration`, consulte Paso 6: ver el historial de revisiones de un documento.

Paso 6: ver el historial de revisiones de un documento

Tras modificar los datos de registro de un vehículo en el paso anterior, puede consultar el historial de todos sus propietarios registrados y cualquier otro campo actualizado. En este paso, consulta el historial de revisiones de un documento de la tabla `VehicleRegistration` del libro mayor `vehicle-registration`.

Para ver el historial de revisiones

1. Revise el siguiente programa (`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**
>
> - Puede ver el historial de revisiones de un documento consultando Función de historial integrada en la siguiente sintaxis.

```
SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

- Tanto la hora de inicio como la hora de finalización son opcionales. Son valores literales de Amazon Ion que se pueden indicar con acentos graves (`` `...` ``). Para obtener más información, consulte Consulta de Ion con PartiQL en Amazon QLDB.
- Como práctica recomendada, califique una consulta de historial con un intervalo de fechas (hora de inicio y hora de finalización) y un identificador de documento (`metadata.id`). QLDB procesa las consultas `SELECT` en las transacciones, que están sujetas a un límite de tiempo de espera de las transacciones.

  El historial de QLDB se indexa por ID de documento y no se pueden crear índices de historial adicionales en este momento. Las consultas de historial que incluyen una hora de inicio y una hora de finalización se benefician de la calificación por intervalo de fechas.

2. Compile y ejecute el programa `QueryHistory.java` para consultar el historial de revisiones del documento `VehicleRegistration` con el VIN `1N4AL11D75C109151`.

Para verificar criptográficamente la revisión de un documento en el libro mayor `vehicle-registration`, continúe con Paso 7: verificar un documento en un libro mayor.

Paso 7: verificar un documento en un libro mayor

Con Amazon QLDB, puede verificar de manera eficiente la integridad de un documento del diario de su libro mayor mediante el uso de hash criptográfico con SHA-256. Para obtener más información sobre cómo funcionan la verificación y el hash criptográfico en QLDB, consulte Verificación de datos en Amazon QLDB.

En este paso, verificará la revisión de un documento en la tabla `VehicleRegistration` del libro mayor `vehicle-registration`. En primer lugar, solicitará un resumen, que se devuelve como un archivo de salida y actúa como firma de todo el historial de cambios del libro mayor. A continuación, solicita una prueba de la revisión relativa a ese resumen. Con esta prueba, se verifica la integridad de la revisión si se aprueban todas las comprobaciones de validación.

Para verificar la revisión de un documento

1. Revise los siguientes archivos `.java`, que representan los objetos de QLDB necesarios para las clases de verificación y de utilidad, con métodos auxiliares para los valores Ion y de cadenas.

   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. Utilice dos archivos `.java` (`GetDigest.java` y `GetRevision.java`) para completar los siguientes pasos:

   - Solicitar un nuevo resumen del libro mayor `vehicle-registration`.
   - Solicitar una prueba de cada revisión del documento de la tabla `VehicleRegistration`.
   - Verifique las revisiones utilizando el resumen devuelto y compruébelo recalculando el resumen.

   El programa `GetDigest.java` contiene el siguiente código.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> Utilice el método `getDigest` para solicitar un resumen que incluya la sugerencia actual del diario del libro mayor. La sugerencia del diario hace referencia al último bloque ingresado en el momento en que QLDB recibe su solicitud.

El programa `GetRevision.java` contiene el siguiente código.

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**
>
> Una vez que el método `getRevision` devuelve una prueba de la revisión del documento especificada, este programa utiliza una API del lado del cliente para verificar esa revisión. Para obtener una descripción general del algoritmo utilizado por esta API, consulte Uso de una prueba para volver a calcular el resumen.

3. Compile y ejecute el programa `GetRevision.java` para verificar criptográficamente el documento `VehicleRegistration` con el VIN `1N4AL11D75C109151`.

Para exportar y validar los datos del diario en el libro mayor `vehicle-registration`, continúe con Paso 8: exportar y validar los datos del diario en un libro mayor.

Paso 8: exportar y validar los datos del diario en un libro mayor

En Amazon QLDB, puede acceder al contenido del diario de su libro mayor para diversos fines, como la retención de datos, el análisis y la auditoría. Para obtener más información, consulte Exportación de datos de diarios desde Amazon QLDB.

En este paso, exporta bloques de diario del libro mayor `vehicle-registration` a un bucket de Amazon S3. A continuación, utiliza los datos exportados para validar la cadena de hash entre los bloques de diario y los componentes de hash individuales de cada bloque.

La entidad principal (IAM) AWS Identity and Access Management que utilice debe tener permisos de IAM suficientes para crear un bucket de Amazon S3 en su Cuenta de AWS. Para obtener información, consulte políticas y permisos en Amazon S3 en la Guía del usuario de Amazon S3. También debe tener permisos para crear un rol de IAM con una política de permisos adjunta que permita a QLDB escribir objetos en su bucket de Amazon S3. Para obtener más información, consulte permisos necesarios para acceder a los recursos de IAM en la Guía del usuario de IAM.

Exportar y validar los datos del diario

1. Revise el siguiente archivo (`JournalBlock.java`), que representa un bloque de diario y su contenido de datos. Incluye un método denominado `verifyBlockHash()` que muestra cómo calcular cada componente individual de un hash de bloque.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

package software.amazon.qldb.tutorial.qldb;

import com.fasterxml.jackson.annotation.JsonCreator;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.fasterxml.jackson.dataformat.ion.IonTimestampSerializers;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.qldb.tutorial.Constants;
import software.amazon.qldb.tutorial.Verifier;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.Date;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

import static java.nio.ByteBuffer.wrap;
```

```
/**
 * Represents a JournalBlock that was recorded after executing a transaction
 * in the ledger.
 */
public final class JournalBlock {
    private static final Logger log = LoggerFactory.getLogger(JournalBlock.class);

    private BlockAddress blockAddress;
    private String transactionId;
    @JsonSerialize(using =
 IonTimestampSerializers.IonTimestampJavaDateSerializer.class)
    private Date blockTimestamp;
    private byte[] blockHash;
    private byte[] entriesHash;
    private byte[] previousBlockHash;
    private byte[][] entriesHashList;
    private TransactionInfo transactionInfo;
    private RedactionInfo redactionInfo;
    private List<QldbRevision> revisions;

    @JsonCreator
    public JournalBlock(@JsonProperty("blockAddress") final BlockAddress
 blockAddress,
                        @JsonProperty("transactionId") final String transactionId,
                        @JsonProperty("blockTimestamp") final Date blockTimestamp,
                        @JsonProperty("blockHash") final byte[] blockHash,
                        @JsonProperty("entriesHash") final byte[] entriesHash,
                        @JsonProperty("previousBlockHash") final byte[]
 previousBlockHash,
                        @JsonProperty("entriesHashList") final byte[][]
 entriesHashList,
                        @JsonProperty("transactionInfo") final TransactionInfo
 transactionInfo,
                        @JsonProperty("redactionInfo") final RedactionInfo
 redactionInfo,
                        @JsonProperty("revisions") final List<QldbRevision>
 revisions) {
        this.blockAddress = blockAddress;
        this.transactionId = transactionId;
        this.blockTimestamp = blockTimestamp;
        this.blockHash = blockHash;
        this.entriesHash = entriesHash;
        this.previousBlockHash = previousBlockHash;
```

```
        this.entriesHashList = entriesHashList;
        this.transactionInfo = transactionInfo;
        this.redactionInfo = redactionInfo;
        this.revisions = revisions;
    }

    public BlockAddress getBlockAddress() {
        return blockAddress;
    }

    public String getTransactionId() {
        return transactionId;
    }

    public Date getBlockTimestamp() {
        return blockTimestamp;
    }

    public byte[][] getEntriesHashList() {
        return entriesHashList;
    }

    public TransactionInfo getTransactionInfo() {
        return transactionInfo;
    }

    public RedactionInfo getRedactionInfo() {
        return redactionInfo;
    }

    public List<QldbRevision> getRevisions() {
        return revisions;
    }

    public byte[] getEntriesHash() {
        return entriesHash;
    }

    public byte[] getBlockHash() {
        return blockHash;
    }

    public byte[] getPreviousBlockHash() {
        return previousBlockHash;
```

```
    }

    @Override
    public String toString() {
        return "JournalBlock{"
                + "blockAddress=" + blockAddress
                + ", transactionId='" + transactionId + '\''
                + ", blockTimestamp=" + blockTimestamp
                + ", blockHash=" + Arrays.toString(blockHash)
                + ", entriesHash=" + Arrays.toString(entriesHash)
                + ", previousBlockHash=" + Arrays.toString(previousBlockHash)
                + ", entriesHashList=" + Arrays.toString(entriesHashList)
                + ", transactionInfo=" + transactionInfo
                + ", redactionInfo=" + redactionInfo
                + ", revisions=" + revisions
                + '}';
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof JournalBlock)) {
            return false;
        }

        final JournalBlock that = (JournalBlock) o;

        if (!getBlockAddress().equals(that.getBlockAddress())) {
            return false;
        }
        if (!getTransactionId().equals(that.getTransactionId())) {
            return false;
        }
        if (!getBlockTimestamp().equals(that.getBlockTimestamp())) {
            return false;
        }
        if (!Arrays.equals(getBlockHash(), that.getBlockHash())) {
            return false;
        }
        if (!Arrays.equals(getEntriesHash(), that.getEntriesHash())) {
            return false;
        }
```

```
        if (!Arrays.equals(getPreviousBlockHash(), that.getPreviousBlockHash())) {
            return false;
        }
        if (!Arrays.deepEquals(getEntriesHashList(), that.getEntriesHashList())) {
            return false;
        }
        if (!getTransactionInfo().equals(that.getTransactionInfo())) {
            return false;
        }
        if (getRedactionInfo() != null ? !
getRedactionInfo().equals(that.getRedactionInfo()) : that.getRedactionInfo() !=
 null) {
            return false;
        }
        return getRevisions() != null ?
 getRevisions().equals(that.getRevisions()) : that.getRevisions() == null;
    }

    @Override
    public int hashCode() {
        int result = getBlockAddress().hashCode();
        result = 31 * result + getTransactionId().hashCode();
        result = 31 * result + getBlockTimestamp().hashCode();
        result = 31 * result + Arrays.hashCode(getBlockHash());
        result = 31 * result + Arrays.hashCode(getEntriesHash());
        result = 31 * result + Arrays.hashCode(getPreviousBlockHash());
        result = 31 * result + Arrays.deepHashCode(getEntriesHashList());
        result = 31 * result + getTransactionInfo().hashCode();
        result = 31 * result + (getRedactionInfo() != null ?
 getRedactionInfo().hashCode() : 0);
        result = 31 * result + (getRevisions() != null ?
 getRevisions().hashCode() : 0);
        return result;
    }

    /**
     * This method validates that the hashes of the components of a journal block
 make up the block
     * hash that is provided with the block itself.
     *
     * The components that contribute to the hash of the journal block consist of
 the following:
     *   - user transaction information (contained in [transactionInfo])
     *   - user redaction information (contained in [redactionInfo])
```

```
    *   - user revisions (contained in [revisions])
    *   - hashes of internal-only system metadata (contained in [revisions] and in
[entriesHashList])
    *   - the previous block hash
    *
    * If any of the computed hashes of user information cannot be validated or any
of the system
    * hashes do not result in the correct computed values, this method will throw
an IllegalArgumentException.
    *
    * Internal-only system metadata is represented by its hash, and can be present
in the form of certain
    * items in the [revisions] list that only contain a hash and no user data, as
well as some hashes
    * in [entriesHashList].
    *
    * To validate that the hashes of the user data are valid components of the
[blockHash], this method
    * performs the following steps:
    *
    * 1. Compute the hash of the [transactionInfo] and validate that it is
included in the [entriesHashList].
    * 2. Compute the hash of the [redactionInfo], if present, and validate that it
is included in the [entriesHashList].
    * 3. Validate the hash of each user revision was correctly computed and
matches the hash published
    * with that revision.
    * 4. Compute the hash of the [revisions] by treating the revision hashes as
the leaf nodes of a Merkle tree
    * and calculating the root hash of that tree. Then validate that hash is
included in the [entriesHashList].
    * 5. Compute the hash of the [entriesHashList] by treating the hashes as the
leaf nodes of a Merkle tree
    * and calculating the root hash of that tree. Then validate that hash matches
[entriesHash].
    * 6. Finally, compute the block hash by computing the hash resulting from
concatenating the [entriesHash]
    * and previous block hash, and validate that the result matches the
[blockHash] provided by QLDB with the block.
    *
    * This method is called by ValidateQldbHashChain::verify for each journal
block to validate its
    * contents before verifying that the hash chain between consecutive blocks is
correct.
```

```
     */
    public void verifyBlockHash() {
        Set<ByteBuffer> entriesHashSet = new HashSet<>();
        Arrays.stream(entriesHashList).forEach(hash ->
 entriesHashSet.add(wrap(hash).asReadOnlyBuffer()));

        byte[] computedTransactionInfoHash = computeTransactionInfoHash();
        if (!
entriesHashSet.contains(wrap(computedTransactionInfoHash).asReadOnlyBuffer())) {
            throw new IllegalArgumentException(
                    "Block transactionInfo hash is not contained in the QLDB block
 entries hash list.");
        }

        if (redactionInfo != null) {
            byte[] computedRedactionInfoHash = computeRedactionInfoHash();
            if (!
entriesHashSet.contains(wrap(computedRedactionInfoHash).asReadOnlyBuffer())) {
                throw new IllegalArgumentException(
                        "Block redactionInfo hash is not contained in the QLDB
 block entries hash list.");
            }
        }

        if (revisions != null) {
            revisions.forEach(QldbRevision::verifyRevisionHash);
            byte[] computedRevisionsHash = computeRevisionsHash();
            if (!
entriesHashSet.contains(wrap(computedRevisionsHash).asReadOnlyBuffer())) {
                throw new IllegalArgumentException(
                        "Block revisions list hash is not contained in the QLDB
 block entries hash list.");
            }
        }

        byte[] computedEntriesHash = computeEntriesHash();
        if (!Arrays.equals(computedEntriesHash, entriesHash)) {
            throw new IllegalArgumentException("Computed entries hash does not
 match entries hash provided in the block.");
        }

        byte[] computedBlockHash = Verifier.dot(computedEntriesHash,
 previousBlockHash);
        if (!Arrays.equals(computedBlockHash, blockHash)) {
```

```
            throw new IllegalArgumentException("Computed block hash does not match
 block hash provided in the block.");
        }
    }

    private byte[] computeTransactionInfoHash() {
        try {
            return
 QldbIonUtils.hashIonValue(Constants.MAPPER.writeValueAsIonValue(transactionInfo));
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not compute transactionInfo
 hash to verify block hash.", e);
        }
    }

    private byte[] computeRedactionInfoHash() {
        try {
            return
 QldbIonUtils.hashIonValue(Constants.MAPPER.writeValueAsIonValue(redactionInfo));
        } catch (IOException e) {
            throw new IllegalArgumentException("Could not compute redactionInfo
 hash to verify block hash.", e);
        }
    }

    private byte[] computeRevisionsHash() {
        return
 Verifier.calculateMerkleTreeRootHash(revisions.stream().map(QldbRevision::getHash).collect
    }

    private byte[] computeEntriesHash() {
        return
 Verifier.calculateMerkleTreeRootHash(Arrays.asList(entriesHashList));
    }
}
```

2. Compile y ejecute el siguiente programa (`ValidateQldbHashChain.java`) para completar los siguientes pasos:

    1. Exporte los bloques del diario del libro mayor `vehicle-registration` a un bucket de Amazon S3 con el nombre **`qldb-tutorial-journal-export-`*`111122223333`*** (sustitúyalos por su número Cuenta de AWS).

2. Valide los componentes de hash individuales de cada bloque llamando a `verifyBlockHash()`.
3. Valide la cadena de hash entre los bloques de diario.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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;
        }
    }

}
```

Si ya no necesita usar el libro mayor `vehicle-registration`, continúe con Paso 9 (opcional): limpiar recursos.

Paso 9 (opcional): limpiar recursos

Puede seguir utilizando el libro mayor `vehicle-registration`. Sin embargo, si ya no lo necesita, debe eliminarlo.

Para eliminar el libro mayor

1. Compile y ejecute el siguiente programa (`DeleteLedger.java`) para eliminar el libro mayor `vehicle-registration` y todo su contenido.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
```

```
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> Si la protección contra eliminación está habilitada para su libro mayor, primero debe desactivarla para poder eliminar el libro mayor utilizando la API de QLDB.

2. Si exportó los datos del diario en el paso anterior y ya no los necesita, utilice la consola de Amazon S3 para eliminar el bucket de S3.

   Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3.

Tutorial de Amazon QLDB Node.js

En esta implementación de la aplicación de ejemplo del tutorial, utilizará el controlador Amazon QLDB con el SDK de AWS para JavaScript en Node.js para crear un libro mayor de QLDB y rellenarlo con datos de ejemplo.

Mientras realiza este tutorial, puede consultar la Referencia de la API de AWS SDK for JavaScript. Para las operaciones de datos transaccionales, puede consultar la referencia del controlador QLDB para la API Node.js.

> **Note**
>
> Cuando proceda, algunos pasos del tutorial incluyen comandos o ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Node.js.

Temas

Instalación de la aplicación de ejemplo Amazon QLDB para Node.js

En esta sección se describe cómo instalar y ejecutar la aplicación de ejemplo de Amazon QLDB proporcionada para este tutorial de Node.js paso a paso. El caso de uso de esta aplicación de ejemplo es una base de datos del Departamento de Vehículos Automóviles (DMV) que rastrea la información histórica completa de las matriculaciones de vehículos.

Esta aplicación de ejemplo de DMV para Node.js es de código abierto y se encuentra en el repositorio de GitHub aws-samples/amazon-qldb-dmv-sample-nodejs.

Requisitos previos

Antes de comenzar, asegúrese de haber completado la configuración del controlador QLDB para Node.js Requisitos previos. Esto incluye instalar Node.js y hacer lo siguiente:

1. Regístrese en AWS.
2. Cree un usuario con los permisos de QLDB adecuados.
3. Conceda acceso programático de desarrollo.

Para completar todos los pasos de este tutorial, necesitará acceso administrativo completo a su recurso de libro mayor a través de la API de QLDB.

Instalación

Para instalar la aplicación de muestra

1. Introduzca el siguiente comando para clonar la aplicación de muestra desde GitHub.

   2.x

   ```
   git clone https://github.com/aws-samples/amazon-qldb-dmv-sample-nodejs.git
   ```

   1.x

   ```
   git clone -b v1.0.0 https://github.com/aws-samples/amazon-qldb-dmv-sample-nodejs.git
   ```

   La aplicación de ejemplo empaqueta el código origen completo de este tutorial y sus dependencias, incluidos el controlador Node.js y el SDK de AWS para JavaScript en Node.js. Esta aplicación está escrita en TypeScript.

2. Cambie al directorio en el que está clonado el paquete `amazon-qldb-dmv-sample-nodejs`.

   ```
   cd amazon-qldb-dmv-sample-nodejs
   ```

3. Realice una instalación limpia de las dependencias.

   ```
   npm ci
   ```

4. Transpile el paquete.

   ```
   npm run build
   ```

   Los archivos JavaScript transpilados se escriben en el directorio `./dist`.

5. Continúe con Paso 1: crear un nuevo libro mayor para iniciar el tutorial y crear un libro mayor.

Paso 1: crear un nuevo libro mayor

En este paso, creará un nuevo libro de mayor de Amazon QLDB denominado `vehicle-registration`.

Para crear un nuevo libro mayor

1. Revise el siguiente archivo (`Constants.ts`), que contiene valores constantes que utilizan todos los demás programas de este tutorial.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

/**
 * Constant values used throughout this tutorial.
 */
export const LEDGER_NAME = "vehicle-registration";
export const LEDGER_NAME_WITH_TAGS = "tags";

export const DRIVERS_LICENSE_TABLE_NAME = "DriversLicense";
export const PERSON_TABLE_NAME = "Person";
```

```
export const VEHICLE_REGISTRATION_TABLE_NAME = "VehicleRegistration";
export const VEHICLE_TABLE_NAME = "Vehicle";

export const GOV_ID_INDEX_NAME = "GovId";
export const LICENSE_NUMBER_INDEX_NAME = "LicenseNumber";
export const LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber";
export const PERSON_ID_INDEX_NAME = "PersonId";
export const VIN_INDEX_NAME = "VIN";

export const RETRY_LIMIT = 4;

export const JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-tutorial-journal-export";
export const USER_TABLES = "information_schema.user_tables";
```

2. Utilice el siguiente programa (`CreateLedger.ts`) para crear un libro mayor denominado `vehicle-registration`.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QLDB } from "aws-sdk";
```

```
import {
    CreateLedgerRequest,
    CreateLedgerResponse,
    DescribeLedgerRequest,
    DescribeLedgerResponse
} from "aws-sdk/clients/qldb";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { sleep } from "./qldb/Util";

const LEDGER_CREATION_POLL_PERIOD_MS = 10000;
const ACTIVE_STATE = "ACTIVE";

/**
 * Create a new ledger with the specified name.
 * @param ledgerName Name of the ledger to be created.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a CreateLedgerResponse.
 */
export async function createLedger(ledgerName: string, qldbClient: QLDB):
 Promise<CreateLedgerResponse> {
    log(`Creating a ledger named: ${ledgerName}...`);
    const request: CreateLedgerRequest = {
        Name: ledgerName,
        PermissionsMode: "ALLOW_ALL"
    }
    const result: CreateLedgerResponse = await
 qldbClient.createLedger(request).promise();
    log(`Success. Ledger state: ${result.State}.`);
    return result;
}

/**
 * Wait for the newly created ledger to become active.
 * @param ledgerName Name of the ledger to be checked on.
 * @param qldbClient The QLDB control plane client to use.
 * @returns Promise which fulfills with a DescribeLedgerResponse.
 */
export async function waitForActive(ledgerName: string, qldbClient: QLDB):
 Promise<DescribeLedgerResponse> {
    log(`Waiting for ledger ${ledgerName} to become active...`);
    const request: DescribeLedgerRequest = {
        Name: ledgerName
```

```
    }
    while (true) {
        const result: DescribeLedgerResponse = await
 qldbClient.describeLedger(request).promise();
        if (result.State === ACTIVE_STATE) {
            log("Success. Ledger is active and ready to be used.");
            return result;
        }
        log("The ledger is still creating. Please wait...");
        await sleep(LEDGER_CREATION_POLL_PERIOD_MS);
    }
}

/**
 * Create a ledger and wait for it to be active.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbClient: QLDB = new QLDB();
        await createLedger(LEDGER_NAME, qldbClient);
        await waitForActive(LEDGER_NAME, qldbClient);
    } catch (e) {
        error(`Unable to create the ledger: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> - En la llamada `createLedger`, debe especificar un nombre de libro mayor y un modo de permisos. Recomendamos encarecidamente el modo de permisos `STANDARD` para maximizar la seguridad de los datos del libro mayor.
> - Al crear un libro mayor, se habilita de forma predeterminada la protección contra la eliminación. Se trata de una característica de QLDB que impide que los libros mayores sean eliminados por cualquier usuario. Tiene la opción de deshabilitar la protección contra la eliminación al crear el libro mayor con la API QLDB o AWS Command Line Interface (AWS CLI).

- Si lo desea, también puede especificar etiquetas para adjuntar al libro mayor.

3. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/CreateLedger.js
```

Para comprobar la conexión con el nuevo libro mayor, continúe con Paso 2: probar la conectividad con el libro mayor.

Paso 2: probar la conectividad con el libro mayor

En este paso, verifica que puede conectarse al libro mayor `vehicle-registration` de Amazon QLDB mediante el punto de conexión de la API de datos transaccionales.

Para probar la conectividad con el libro mayor

1. Utilice el siguiente programa (`ConnectToLedger.ts`) para crear una conexión de sesión de datos al libro mayor `vehicle-registration`.

   2.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
```

```
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, RetryConfig  } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/qldbsession";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

const qldbDriver: QldbDriver = createQldbDriver();

/**
 * Create a driver for creating sessions.
 * @param ledgerName The name of the ledger to create the driver on.
 * @param serviceConfigurationOptions The configurations for the AWS SDK client
 that the driver uses.
 * @returns The driver for creating sessions.
 */
export function createQldbDriver(
    ledgerName: string = LEDGER_NAME,
    serviceConfigurationOptions: ClientConfiguration = {}
): QldbDriver {
    const retryLimit = 4;
    const maxConcurrentTransactions = 10;
    //Use driver's default backoff function (and hence, no second parameter
 provided to RetryConfig)
    const retryConfig: RetryConfig = new RetryConfig(retryLimit);
    const qldbDriver: QldbDriver = new QldbDriver(ledgerName,
 serviceConfigurationOptions, maxConcurrentTransactions, retryConfig);
    return qldbDriver;
}

export function getQldbDriver(): QldbDriver {
    return qldbDriver;
}

/**
 * Connect to a session for a given ledger using default settings.
 * @returns Promise which fulfills with void.
 */
```

```
const main = async function(): Promise<void> {
    try {
        log("Listing table names...");
        const tableNames: string[] = await qldbDriver.getTableNames();
        tableNames.forEach((tableName: string): void => {
            log(tableName);
        });
    } catch (e) {
        error(`Unable to create session: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

1.x

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
 THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
```

```
import { QldbDriver  } from "amazon-qldb-driver-nodejs";
import { ClientConfiguration } from "aws-sdk/clients/qldbsession";

import { LEDGER_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

const qldbDriver: QldbDriver = createQldbDriver();

/**
 * Create a driver for creating sessions.
 * @param ledgerName The name of the ledger to create the driver on.
 * @param serviceConfigurationOptions The configurations for the AWS SDK client
 that the driver uses.
 * @returns The driver for creating sessions.
 */
export function createQldbDriver(
    ledgerName: string = LEDGER_NAME,
    serviceConfigurationOptions: ClientConfiguration = {}
): QldbDriver {
    const qldbDriver: QldbDriver = new QldbDriver(ledgerName,
 serviceConfigurationOptions);
    return qldbDriver;
}

export function getQldbDriver(): QldbDriver {
    return qldbDriver;
}

/**
 * Connect to a session for a given ledger using default settings.
 * @returns Promise which fulfills with void.
 */
var main = async function(): Promise<void> {
    try {
        log("Listing table names...");
        const tableNames: string[] = await qldbDriver.getTableNames();
        tableNames.forEach((tableName: string): void => {
            log(tableName);
        });
    } catch (e) {
        error(`Unable to create session: ${e}`);
    }
}
```

```
if (require.main === module) {
    main();
}
```

> **Note**
>
> Para ejecutar transacciones de datos en su libro mayor, debe crear un objeto de controlador de QLDB para conectarse a un libro mayor específico. Se trata de un objeto de cliente diferente al objeto `qldbClient` que utilizó en el paso anterior para crear el libro mayor. Ese cliente anterior solo se usa para ejecutar las operaciones de la API de administración que se enumeran en Referencia de la API de Amazon QLDB.

2. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/ConnectToLedger.js
```

Para crear tablas en el libro mayor `vehicle-registration`, continúe con Paso 3: cree tablas, índices y datos de muestra.

Paso 3: cree tablas, índices y datos de muestra

Cuando su libro mayor de Amazon QLDB esté activo y acepte conexiones, podrá empezar a crear tablas con datos sobre los vehículos, sus propietarios y su información de registro. Tras crear las tablas y los índices, puede cargarlos con datos.

En este paso, creará cuatro tablas en el libro mayor `vehicle-registration`:

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

También se crean los siguientes índices.

Nombre de la tabla	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Al insertar datos de ejemplo, primero debe insertar los documentos en la tabla `Person`. A continuación, utilice los `id` asignados por el sistema de cada documento `Person` para rellenar los campos pertinentes en los documentos `VehicleRegistration` y `DriversLicense` correspondientes.

> **Tip**
>
> Como práctica recomendada, utilice un `id` de documento asignado por el sistema como clave externa. Si bien puede definir campos que pretenden ser identificadores únicos (por ejemplo, el VIN de un vehículo), el verdadero identificador único de un documento es su `id`. Este campo se incluye en los metadatos del documento, que puede consultar en la vista confirmada (la vista de una tabla definida por el sistema).
> Para obtener más información acerca de las vistas en QLDB, consulte Conceptos clave.
> Para obtener más información sobre metadatos, consulte Consulta de los metadatos del documento.

Para crear tablas e índices

1. Utilice el siguiente programa (`CreateTable.ts`) para crear las tablas mencionadas anteriormente.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";

import { getQldbDriver } from "./ConnectToLedger";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Create multiple tables in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to create.
 * @returns Promise which fulfills with the number of changes to the database.
 */
export async function createTable(txn: TransactionExecutor, tableName: string):
 Promise<number> {
    const statement: string = `CREATE TABLE ${tableName}`;
```

```
    return await txn.execute(statement).then((result: Result) => {
        log(`Successfully created table ${tableName}.`);
        return result.getResultList().length;
    });
}

/**
 * Create tables in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            Promise.all([
                createTable(txn, VEHICLE_REGISTRATION_TABLE_NAME),
                createTable(txn, VEHICLE_TABLE_NAME),
                createTable(txn, PERSON_TABLE_NAME),
                createTable(txn, DRIVERS_LICENSE_TABLE_NAME)
            ]);
        });
    } catch (e) {
        error(`Unable to create tables: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

> **Note**
>
> Este programa muestra cómo utilizar la función `executeLambda` en una instancia de controlador QLDB. En este ejemplo, ejecuta varias instrucciones `CREATE TABLE` de PartiQL con una sola expresión lambda.
> Esta función de ejecución inicia implícitamente una transacción, ejecuta todas las instrucciones de lambda y, a continuación, confirma automáticamente la transacción.

2. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/CreateTable.js
```

3. Utilice el siguiente programa (`CreateIndex.ts`) para crear índices en las tablas, tal y como se describió anteriormente.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, TransactionExecutor } from "amazon-qldb-driver-nodejs";

import { getQldbDriver } from "./ConnectToLedger";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    GOV_ID_INDEX_NAME,
    LICENSE_NUMBER_INDEX_NAME,
    LICENSE_PLATE_NUMBER_INDEX_NAME,
    PERSON_ID_INDEX_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME,
    VIN_INDEX_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
```

```
/**
 * Create an index for a particular table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to add indexes for.
 * @param indexAttribute Index to create on a single attribute.
 * @returns Promise which fulfills with the number of changes to the database.
 */
export async function createIndex(
    txn: TransactionExecutor,
    tableName: string,
    indexAttribute: string
): Promise<number> {
    const statement: string = `CREATE INDEX on ${tableName} (${indexAttribute})`;
    return await txn.execute(statement).then((result) => {
        log(`Successfully created index ${indexAttribute} on table ${tableName}.`);
        return result.getResultList().length;
    });
}

/**
 * Create indexes on tables in a particular ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            Promise.all([
                createIndex(txn, PERSON_TABLE_NAME, GOV_ID_INDEX_NAME),
                createIndex(txn, VEHICLE_TABLE_NAME, VIN_INDEX_NAME),
                createIndex(txn, VEHICLE_REGISTRATION_TABLE_NAME, VIN_INDEX_NAME),
                createIndex(txn, VEHICLE_REGISTRATION_TABLE_NAME,
 LICENSE_PLATE_NUMBER_INDEX_NAME),
                createIndex(txn, DRIVERS_LICENSE_TABLE_NAME, PERSON_ID_INDEX_NAME),
                createIndex(txn, DRIVERS_LICENSE_TABLE_NAME,
 LICENSE_NUMBER_INDEX_NAME)
            ]);
        });
    } catch (e) {
        error(`Unable to create indexes: ${e}`);
    }
}
```

```
if (require.main === module) {
    main();
}
```

4. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/CreateIndex.js
```

Para cargar los datos en las tablas

1. Revise los siguientes archivos `.ts`.

   1. `SampleData.ts`: contiene los datos de ejemplo que se insertan en las tablas `vehicle-registration`.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Decimal } from "ion-js";
```

```
const EMPTY_SECONDARY_OWNERS: object[] = [];
export const DRIVERS_LICENSE = [
    {
        PersonId: "",
        LicenseNumber: "LEWISR261LL",
        LicenseType: "Learner",
        ValidFromDate: new Date("2016-12-20"),
        ValidToDate: new Date("2020-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "LOGANB486CG",
        LicenseType: "Probationary",
        ValidFromDate : new Date("2016-04-06"),
        ValidToDate : new Date("2020-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "744 849 301",
        LicenseType: "Full",
        ValidFromDate : new Date("2017-12-06"),
        ValidToDate : new Date("2022-10-15")
    },
    {
        PersonId: "",
        LicenseNumber : "P626-168-229-765",
        LicenseType: "Learner",
        ValidFromDate : new Date("2017-08-16"),
        ValidToDate : new Date("2021-11-15")
    },
    {
        PersonId: "",
        LicenseNumber : "S152-780-97-415-0",
        LicenseType: "Probationary",
        ValidFromDate : new Date("2015-08-15"),
        ValidToDate : new Date("2021-08-21")
    }
];
export const PERSON = [
    {
        FirstName : "Raul",
        LastName : "Lewis",
        DOB : new Date("1963-08-19"),
        Address : "1719 University Street, Seattle, WA, 98109",
```

```
        GovId : "LEWISR261LL",
        GovIdType : "Driver License"
    },
    {
        FirstName : "Brent",
        LastName : "Logan",
        DOB : new Date("1967-07-03"),
        Address : "43 Stockert Hollow Road, Everett, WA, 98203",
        GovId : "LOGANB486CG",
        GovIdType : "Driver License"
    },
    {
        FirstName : "Alexis",
        LastName : "Pena",
        DOB : new Date("1974-02-10"),
        Address : "4058 Melrose Street, Spokane Valley, WA, 99206",
        GovId : "744 849 301",
        GovIdType : "SSN"
    },
    {
        FirstName : "Melvin",
        LastName : "Parker",
        DOB : new Date("1976-05-22"),
        Address : "4362 Ryder Avenue, Seattle, WA, 98101",
        GovId : "P626-168-229-765",
        GovIdType : "Passport"
    },
    {
        FirstName : "Salvatore",
        LastName : "Spencer",
        DOB : new Date("1997-11-15"),
        Address : "4450 Honeysuckle Lane, Seattle, WA, 98101",
        GovId : "S152-780-97-415-0",
        GovIdType : "Passport"
    }
];
export const VEHICLE = [
    {
        VIN : "1N4AL11D75C109151",
        Type : "Sedan",
        Year : 2011,
        Make : "Audi",
        Model : "A5",
        Color : "Silver"
```

```
    },
    {
        VIN : "KM8SRDHF6EU074761",
        Type : "Sedan",
        Year : 2015,
        Make : "Tesla",
        Model : "Model S",
        Color : "Blue"
    },
    {
        VIN : "3HGGK5G53FM761765",
        Type : "Motorcycle",
        Year : 2011,
        Make : "Ducati",
        Model : "Monster 1200",
        Color : "Yellow"
    },
    {
        VIN : "1HVBBAANXWH544237",
        Type : "Semi",
        Year : 2009,
        Make : "Ford",
        Model : "F 150",
        Color : "Black"
    },
    {
        VIN : "1C4RJFAG0FC625797",
        Type : "Sedan",
        Year : 2019,
        Make : "Mercedes",
        Model : "CLK 350",
        Color : "White"
    }
];
export const VEHICLE_REGISTRATION = [
    {
        VIN : "1N4AL11D75C109151",
        LicensePlateNumber : "LEWISR261LL",
        State : "WA",
        City : "Seattle",
        ValidFromDate : new Date("2017-08-21"),
        ValidToDate : new Date("2020-05-11"),
        PendingPenaltyTicketAmount : new Decimal(9025, -2),
        Owners : {
```

```
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "KM8SRDHF6EU074761",
        LicensePlateNumber : "CA762X",
        State : "WA",
        City : "Kent",
        PendingPenaltyTicketAmount : new Decimal(13075, -2),
        ValidFromDate : new Date("2017-09-14"),
        ValidToDate : new Date("2020-06-25"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "3HGGK5G53FM761765",
        LicensePlateNumber : "CD820Z",
        State : "WA",
        City : "Everett",
        PendingPenaltyTicketAmount : new Decimal(44230, -2),
        ValidFromDate : new Date("2011-03-17"),
        ValidToDate : new Date("2021-03-24"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
        VIN : "1HVBBAANXWH544237",
        LicensePlateNumber : "LS477D",
        State : "WA",
        City : "Tacoma",
        PendingPenaltyTicketAmount : new Decimal(4220, -2),
        ValidFromDate : new Date("2011-10-26"),
        ValidToDate : new Date("2023-09-25"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    },
    {
```

```
        VIN : "1C4RJFAG0FC625797",
        LicensePlateNumber : "TH393F",
        State : "WA",
        City : "Olympia",
        PendingPenaltyTicketAmount : new Decimal(3045, -2),
        ValidFromDate : new Date("2013-09-02"),
        ValidToDate : new Date("2024-03-19"),
        Owners : {
            PrimaryOwner : { PersonId : "" },
            SecondaryOwners : EMPTY_SECONDARY_OWNERS
        }
    }
];
```

2. `Util.ts`: un módulo de utilidad que se importa del paquete `ion-js` para proporcionar funciones auxiliares que convierten, analizan e imprimen datos de Amazon Ion.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { Result, TransactionExecutor } from "amazon-qldb-driver-nodejs";
```

```
import { GetBlockResponse, GetDigestResponse, ValueHolder } from "aws-sdk/
clients/qldb";
import {
    Decimal,
    decodeUtf8,
    dom,
    IonTypes,
    makePrettyWriter,
    makeReader,
    Reader,
    Timestamp,
    toBase64,
    Writer
} from "ion-js";

import { error } from "./LogUtil";



/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given BlockResponse.
 * @param blockResponse The BlockResponse to convert to string.
 * @returns The string representation of the supplied BlockResponse.
 */
export function blockResponseToString(blockResponse: GetBlockResponse): string {
    let stringBuilder: string = "";
    if (blockResponse.Block.IonText) {
        stringBuilder = stringBuilder + "Block: " + blockResponse.Block.IonText +
 ", ";
    }
    if (blockResponse.Proof.IonText) {
        stringBuilder = stringBuilder + "Proof: " + blockResponse.Proof.IonText;
    }
    stringBuilder = "{" + stringBuilder + "}";
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given GetDigestResponse.
```

```
 * @param digestResponse The GetDigestResponse to convert to string.
 * @returns The string representation of the supplied GetDigestResponse.
 */
export function digestResponseToString(digestResponse: GetDigestResponse): string
 {
    let stringBuilder: string = "";
    if (digestResponse.Digest) {
        stringBuilder += "Digest: " + JSON.stringify(toBase64(<Uint8Array>
 digestResponse.Digest)) + ", ";
    }
    if (digestResponse.DigestTipAddress.IonText) {
        stringBuilder += "DigestTipAddress: " +
 digestResponse.DigestTipAddress.IonText;
    }
    stringBuilder = "{" + stringBuilder + "}";
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
    return decodeUtf8(writer.getBytes());
}

/**
 * Get the document IDs from the given table.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName The table name to query.
 * @param field A field to query.
 * @param value The key of the given field.
 * @returns Promise which fulfills with the document ID as a string.
 */
export async function getDocumentId(
    txn: TransactionExecutor,
    tableName: string,
    field: string,
    value: string
): Promise<string> {
    const query: string = `SELECT id FROM ${tableName} AS t BY id WHERE t.
${field} = ?`;
    let documentId: string = undefined;
    await txn.execute(query, value).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to retrieve document ID using ${value}.`);
        }
        documentId = resultList[0].get("id").stringValue();
```

```
    }).catch((err: any) => {
        error(`Error getting documentId: ${err}`);
    });
    return documentId;
}

/**
 * Sleep for the specified amount of time.
 * @param ms The amount of time to sleep in milliseconds.
 * @returns Promise which fulfills with void.
 */
export function sleep(ms: number): Promise<void> {
    return new Promise(resolve => setTimeout(resolve, ms));
}

/**
 * Find the value of a given path in an Ion value. The path should contain a blob
 value.
 * @param value The Ion value that contains the journal block attributes.
 * @param path The path to a certain attribute.
 * @returns Uint8Array value of the blob, or null if the attribute cannot be
 found in the Ion value
 *                  or is not of type Blob
 */
export function getBlobValue(value: dom.Value, path: string): Uint8Array | null {
    const attribute: dom.Value = value.get(path);
    if (attribute !== null && attribute.getType() === IonTypes.BLOB) {
        return attribute.uInt8ArrayValue();
    }
    return null;
}

/**
 * TODO: Replace this with json.stringify
 * Returns the string representation of a given ValueHolder.
 * @param valueHolder The ValueHolder to convert to string.
 * @returns The string representation of the supplied ValueHolder.
 */
export function valueHolderToString(valueHolder: ValueHolder): string {
    const stringBuilder: string = `{ IonText: ${valueHolder.IonText}}`;
    const writer: Writer = makePrettyWriter();
    const reader: Reader = makeReader(stringBuilder);
    writer.writeValues(reader);
```

```
    return decodeUtf8(writer.getBytes());
}

/**
 * Converts a given value to Ion using the provided writer.
 * @param value The value to convert to Ion.
 * @param ionWriter The Writer to pass the value into.
 * @throws Error: If the given value cannot be converted to Ion.
 */
export function writeValueAsIon(value: any, ionWriter: Writer): void {
    switch (typeof value) {
        case "string":
            ionWriter.writeString(value);
            break;
        case "boolean":
            ionWriter.writeBoolean(value);
            break;
        case "number":
                ionWriter.writeInt(value);
                break;
        case "object":
            if (Array.isArray(value)) {
                // Object is an array.
                ionWriter.stepIn(IonTypes.LIST);

                for (const element of value) {
                    writeValueAsIon(element, ionWriter);
                }

                ionWriter.stepOut();
            } else if (value instanceof Date) {
                // Object is a Date.
                ionWriter.writeTimestamp(Timestamp.parse(value.toISOString()));
            } else if (value instanceof Decimal) {
                // Object is a Decimal.
                ionWriter.writeDecimal(value);
            } else if (value === null) {
                ionWriter.writeNull(IonTypes.NULL);
            } else {
                // Object is a struct.
                ionWriter.stepIn(IonTypes.STRUCT);

                for (const key of Object.keys(value)) {
                    ionWriter.writeFieldName(key);
```

```
                    writeValueAsIon(value[key], ionWriter);
                }
                ionWriter.stepOut();
            }
            break;
        default:
            throw new Error(`Cannot convert to Ion for type: ${(typeof
 value)}.`);
    }
}
```

> **Note**
>
> La función `getDocumentId` ejecuta una consulta que devuelve los identificadores de documentos asignados por el sistema desde una tabla. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

2. Utilice el siguiente programa (`InsertDocument.ts`) para insertar los datos de ejemplo en las tablas.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
```

```
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { DRIVERS_LICENSE, PERSON, VEHICLE, VEHICLE_REGISTRATION } from "./model/
SampleData";
import {
    DRIVERS_LICENSE_TABLE_NAME,
    PERSON_TABLE_NAME,
    VEHICLE_REGISTRATION_TABLE_NAME,
    VEHICLE_TABLE_NAME
} from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";

/**
 * Insert the given list of documents into a table in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param tableName Name of the table to insert documents into.
 * @param documents List of documents to insert.
 * @returns Promise which fulfills with a {@linkcode Result} object.
 */
export async function insertDocument(
    txn: TransactionExecutor,
    tableName: string,
    documents: object[]
): Promise<Result> {
    const statement: string = `INSERT INTO ${tableName} ?`;
    const result: Result = await txn.execute(statement, documents);
    return result;
}

/**
 * Handle the insertion of documents and updating PersonIds all in a single
 transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @returns Promise which fulfills with void.
 */
async function updateAndInsertDocuments(txn: TransactionExecutor): Promise<void> {
    log("Inserting multiple documents into the 'Person' table...");
```

```
    const documentIds: Result = await insertDocument(txn, PERSON_TABLE_NAME,
 PERSON);

    const listOfDocumentIds: dom.Value[] = documentIds.getResultList();
    log("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...");
    updatePersonId(listOfDocumentIds);

    log("Inserting multiple documents into the remaining tables...");
    await Promise.all([
        insertDocument(txn, DRIVERS_LICENSE_TABLE_NAME, DRIVERS_LICENSE),
        insertDocument(txn, VEHICLE_REGISTRATION_TABLE_NAME, VEHICLE_REGISTRATION),
        insertDocument(txn, VEHICLE_TABLE_NAME, VEHICLE)
    ]);
}

/**
 * Update the PersonId value for DriversLicense records and the PrimaryOwner value
 for VehicleRegistration records.
 * @param documentIds List of document IDs.
 */
export function updatePersonId(documentIds: dom.Value[]): void {
    documentIds.forEach((value: dom.Value, i: number) => {
        const documentId: string = value.get("documentId").stringValue();
        DRIVERS_LICENSE[i].PersonId = documentId;
        VEHICLE_REGISTRATION[i].Owners.PrimaryOwner.PersonId = documentId;
    });
}

/**
 * Insert documents into a table in a QLDB ledger.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await updateAndInsertDocuments(txn);
        });
    } catch (e) {
        error(`Unable to insert documents: ${e}`);
    }
}
```

```
if (require.main === module) {
    main();
}
```

> **Note**
>
> - Este programa muestra cómo llamar a la función `execute` con valores parametrizados. Puede pasar parámetros de datos además de la instrucción PartiQL que desee ejecutar. Utilice un signo de interrogación (`?`) como marcador de posición variable en la cadena de la instrucción.
> - Si una instrucción `INSERT` tiene éxito, devuelve `id` de cada documento insertado.

3. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/InsertDocument.js
```

A continuación, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`. Continúe en Paso 4: consultar las tablas en un libro mayor.

Paso 4: consultar las tablas en un libro mayor

Tras crear tablas en un libro mayor de Amazon QLDB y cargarlas con datos, puede ejecutar consultas para revisar los datos de registro del vehículo que acaba de insertar. QLDB emplea PartiQL como lenguaje de consulta y Amazon Ion como modelo de datos orientado a documentos.

PartiQL es un lenguaje de consulta de código abierto compatible con SQL que se ha ampliado para funcionar con Ion. PartiQL le permite insertar, consultar y administrar sus datos con operadores SQL conocidos. Amazon Ion es un superconjunto de JSON. Ion es un formato de datos de código abierto basado en documentos que le brinda la flexibilidad de almacenar y procesar datos estructurados, semiestructurados y anidados.

En este paso, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`.

> **Warning**
>
> Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. PartiQL admite este tipo de consultas porque es compatible con SQL.

Sin embargo, no ejecute escaneados de tablas para casos de uso de producción en QLDB. Los escaneos de tablas pueden provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad en un campo indexado o en un ID de documento, por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Para consultar las tablas

1. Utilice el siguiente programa (`FindVehicles.ts`) para consultar todos los vehículos registrados a nombre de una persona en su libro mayor.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> En primer lugar, este programa consulta la tabla `Person` del documento con `GovId` `LEWISR261LL` para obtener su campo de metadatos `id`.
> A continuación, utiliza este `id` de documento como clave externa para consultar la tabla `VehicleRegistration` mediante `PrimaryOwner.PersonId`. También combina `VehicleRegistration` con la tabla `Vehicle` del campo `VIN`.

2. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/FindVehicles.js
```

Para obtener información sobre la modificación de los documentos en las tablas del libro mayor `vehicle-registration`, consulte Paso 5: modificar los documentos de un libro mayor.

Paso 5: modificar los documentos de un libro mayor

Ahora que tiene datos con los que trabajar, puede empezar a realizar cambios en los documentos del libro mayor `vehicle-registration` de Amazon QLDB. En este paso, los siguientes ejemplos de código muestran cómo ejecutar instrucciones de lenguaje de manipulación de datos (DML). Estas instrucciones actualizan al propietario principal de un vehículo y añaden un propietario secundario a otro vehículo.

Para modificar documentos

1. Use el siguiente programa (`TransferVehicleOwnership.ts`) para actualizar el propietario principal del vehículo con VIN `1N4AL11D75C109151` en su libro mayor.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON, VEHICLE } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";

/**
 * Query a driver's information using the given ID.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param documentId The unique ID of a document in the Person table.
 * @returns Promise which fulfills with an Ion value containing the person.
 */
export async function findPersonFromDocumentId(txn: TransactionExecutor,
 documentId: string): Promise<dom.Value> {
    const query: string = "SELECT p.* FROM Person AS p BY pid WHERE pid = ?";

    let personId: dom.Value;
```

```
    await txn.execute(query, documentId).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to find person with ID: ${documentId}.`);
        }
        personId = resultList[0];
    });
    return personId;
}

/**
 * Find the primary owner for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN to find primary owner for.
 * @returns Promise which fulfills with an Ion value containing the primary owner.
 */
export async function findPrimaryOwnerForVehicle(txn: TransactionExecutor, vin:
 string): Promise<dom.Value> {
    log(`Finding primary owner for vehicle with VIN: ${vin}`);
    const query: string = "SELECT Owners.PrimaryOwner.PersonId FROM
 VehicleRegistration AS v WHERE v.VIN = ?";

    let documentId: string = undefined;
    await txn.execute(query, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error(`Unable to retrieve document ID using ${vin}.`);
        }
        const PersonIdValue: dom.Value = resultList[0].get("PersonId");
        if (PersonIdValue === null) {
            throw new Error(`Expected field name PersonId not found.`);
        }
        documentId = PersonIdValue.stringValue();
    });
    return findPersonFromDocumentId(txn, documentId);
}

/**
 * Update the primary owner for a vehicle using the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN for the vehicle to operate on.
 * @param documentId New PersonId for the primary owner.
 * @returns Promise which fulfills with void.
 */
```

```
async function updateVehicleRegistration(txn: TransactionExecutor, vin: string,
 documentId: string): Promise<void> {
    const statement: string = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?";

    log(`Updating the primary owner for vehicle with VIN: ${vin}...`);
    await txn.execute(statement, documentId, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        if (resultList.length === 0) {
            throw new Error("Unable to transfer vehicle, could not find
 registration.");
        }
        log(`Successfully transferred vehicle with VIN ${vin} to new owner.`);
    });
}

/**
 * Validate the current owner of the given vehicle and transfer its ownership to a
 new owner in a single transaction.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin The VIN of the vehicle to transfer ownership of.
 * @param currentOwner The GovId of the current owner of the vehicle.
 * @param newOwner The GovId of the new owner of the vehicle.
 */
export async function validateAndUpdateRegistration(
    txn: TransactionExecutor,
    vin: string,
    currentOwner: string,
    newOwner: string
): Promise<void> {
    const primaryOwner: dom.Value = await findPrimaryOwnerForVehicle(txn, vin);
    const govIdValue: dom.Value = primaryOwner.get("GovId");
    if (govIdValue !== null && govIdValue.stringValue() !== currentOwner) {
        log("Incorrect primary owner identified for vehicle, unable to transfer.");
    }
    else {
        const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME,
 "GovId", newOwner);
        await updateVehicleRegistration(txn, vin, documentId);
        log("Successfully transferred vehicle ownership!");
    }
}

/**
```

```
 * Find primary owner for a particular vehicle's VIN.
 * Transfer to another primary owner for a particular vehicle's VIN.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();

        const vin: string = VEHICLE[0].VIN;
        const previousOwnerGovId: string = PERSON[0].GovId;
        const newPrimaryOwnerGovId: string = PERSON[1].GovId;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            await validateAndUpdateRegistration(txn, vin, previousOwnerGovId,
 newPrimaryOwnerGovId);
        });
    } catch (e) {
        error(`Unable to connect and run queries: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

2. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/TransferVehicleOwnership.js
```

3. Utilice el siguiente programa (`AddSecondaryOwner.ts`) para añadir un propietario secundario al vehículo con el VIN `KM8SRDHF6EU074761` en su libro mayor.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
```

```
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

import { QldbDriver, Result, TransactionExecutor } from "amazon-qldb-driver-
nodejs";
import { dom } from "ion-js";

import { getQldbDriver } from "./ConnectToLedger";
import { PERSON, VEHICLE_REGISTRATION } from "./model/SampleData";
import { PERSON_TABLE_NAME } from "./qldb/Constants";
import { error, log } from "./qldb/LogUtil";
import { getDocumentId } from "./qldb/Util";
import { prettyPrintResultList } from "./ScanTable";

/**
 * Add a secondary owner into 'VehicleRegistration' table for a particular VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN of the vehicle to query.
 * @param secondaryOwnerId The secondary owner's person ID.
 * @returns Promise which fulfills with void.
 */
export async function addSecondaryOwner(
    txn: TransactionExecutor,
    vin: string,
    secondaryOwnerId: string
): Promise<void> {
    log(`Inserting secondary owner for vehicle with VIN: ${vin}`);
    const query: string =
        `FROM VehicleRegistration AS v WHERE v.VIN = ? INSERT INTO
 v.Owners.SecondaryOwners VALUE ?`;

    const personToInsert = {PersonId: secondaryOwnerId};
```

```
    await txn.execute(query, vin, personToInsert).then(async (result: Result) => {
        const resultList: dom.Value[] = result.getResultList();
        log("VehicleRegistration Document IDs which had secondary owners added: ");
        prettyPrintResultList(resultList);
    });
}

/**
 * Query for a document ID with a government ID.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param governmentId The government ID to query with.
 * @returns Promise which fulfills with the document ID as a string.
 */
export async function getDocumentIdByGovId(txn: TransactionExecutor, governmentId:
 string): Promise<string> {
    const documentId: string = await getDocumentId(txn, PERSON_TABLE_NAME, "GovId",
 governmentId);
    return documentId;
}

/**
 * Check whether a driver has already been registered for the given VIN.
 * @param txn The {@linkcode TransactionExecutor} for lambda execute.
 * @param vin VIN of the vehicle to query.
 * @param secondaryOwnerId The secondary owner's person ID.
 * @returns Promise which fulfills with a boolean.
 */
export async function isSecondaryOwnerForVehicle(
    txn: TransactionExecutor,
    vin: string,
    secondaryOwnerId: string
): Promise<boolean> {
    log(`Finding secondary owners for vehicle with VIN: ${vin}`);
    const query: string = "SELECT Owners.SecondaryOwners FROM VehicleRegistration
 AS v WHERE v.VIN = ?";

    let doesExist: boolean = false;

    await txn.execute(query, vin).then((result: Result) => {
        const resultList: dom.Value[] = result.getResultList();

        resultList.forEach((value: dom.Value) => {
            const secondaryOwnersList: dom.Value[] =
 value.get("SecondaryOwners").elements();
```

```
            secondaryOwnersList.forEach((secondaryOwner) => {
                const personId: dom.Value = secondaryOwner.get("PersonId");
                if (personId !== null &&  personId.stringValue() ===
 secondaryOwnerId) {
                    doesExist = true;
                }
            });
        });
    });
    return doesExist;
}

/**
 * Finds and adds secondary owners for a vehicle.
 * @returns Promise which fulfills with void.
 */
const main = async function(): Promise<void> {
    try {
        const qldbDriver: QldbDriver = getQldbDriver();
        const vin: string = VEHICLE_REGISTRATION[1].VIN;
        const govId: string = PERSON[0].GovId;

        await qldbDriver.executeLambda(async (txn: TransactionExecutor) => {
            const documentId: string = await getDocumentIdByGovId(txn, govId);

            if (await isSecondaryOwnerForVehicle(txn, vin, documentId)) {
                log(`Person with ID ${documentId} has already been added as a
 secondary owner of this vehicle.`);
            } else {
                await addSecondaryOwner(txn, vin, documentId);
            }
        });

        log("Secondary owners successfully updated.");
    } catch (e) {
        error(`Unable to add secondary owner: ${e}`);
    }
}

if (require.main === module) {
    main();
}
```

4. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/AddSecondaryOwner.js
```

Para revisar estos cambios en el libro mayor `vehicle-registration`, consulte Paso 6: ver el historial de revisiones de un documento.

Paso 6: ver el historial de revisiones de un documento

Tras modificar los datos de registro de un vehículo en el paso anterior, puede consultar el historial de todos sus propietarios registrados y cualquier otro campo actualizado. En este paso, consulta el historial de revisiones de un documento de la tabla `VehicleRegistration` del libro mayor `vehicle-registration`.

Para ver el historial de revisiones

1. Utilice el siguiente programa (`QueryHistory.ts`) para consultar el historial de revisiones del documento `VehicleRegistration` con el VIN `1N4AL11D75C109151`.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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

- Puede ver el historial de revisiones de un documento consultando la Función de historial integrada en la siguiente sintaxis.

  ```
  SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
  [ WHERE h.metadata.id = 'id' ]
  ```

- Tanto la hora de inicio como la hora de finalización son opcionales. Son valores literales de Amazon Ion que se pueden indicar con acentos graves (`` `...` ``). Para obtener más información, consulte Consulta de Ion con PartiQL en Amazon QLDB.
- Como práctica recomendada, califique una consulta de historial con un intervalo de fechas (hora de inicio y hora de finalización) y un identificador de documento (`metadata.id`). QLDB procesa las consultas `SELECT` en las transacciones, que están sujetas a un límite de tiempo de espera de las transacciones.

  El historial de QLDB se indexa por ID de documento y no se pueden crear índices de historial adicionales en este momento. Las consultas de historial que incluyen una

hora de inicio y una hora de finalización se benefician de la calificación por intervalo de fechas.

2. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/QueryHistory.js
```

Para verificar criptográficamente la revisión de un documento en el libro mayor `vehicle-registration`, continúe con Paso 7: verificar un documento en un libro mayor.

Paso 7: verificar un documento en un libro mayor

Con Amazon QLDB, puede verificar de manera eficiente la integridad de un documento del diario de su libro mayor mediante el uso de hash criptográfico con SHA-256. Para obtener más información sobre cómo funcionan la verificación y el hash criptográfico en QLDB, consulte Verificación de datos en Amazon QLDB.

En este paso, verificará la revisión de un documento en la tabla `VehicleRegistration` del libro mayor `vehicle-registration`. En primer lugar, solicita un resumen, que se devuelve como un archivo de salida y actúa como firma de todo el historial de cambios del libro mayor. A continuación, solicita una prueba de la revisión relativa a ese resumen. Con esta prueba, se verifica la integridad de la revisión si se aprueban todas las comprobaciones de validación.

Para verificar la revisión de un documento

1. Revise los siguientes archivos `.ts`, que contienen los objetos QLDB necesarios para la verificación.

   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. Utilice dos programas `.ts` (`GetDigest.ts` y `GetRevision.ts`) para realizar los siguientes pasos:

   - Solicite un nuevo resumen del libro mayor `vehicle-registration`.
   - Solicite una prueba de cada revisión del documento con el VIN `1N4AL11D75C109151` de la tabla `VehicleRegistration`.
   - Verifique las revisiones utilizando el resumen devuelto y compruébelo recalculando el resumen.

   El programa `GetDigest.ts` contiene el siguiente código.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
```

```
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> Utilice la función `getDigest` para solicitar un resumen que incluya el tip actual del diario del libro mayor. La sugerencia del diario hace referencia al último bloqueo comprometido en el momento en que QLDB recibe su solicitud.

El programa `GetRevision.ts` contiene el siguiente código.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
```

```
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> Una vez que la función `getRevision` devuelve una prueba de la revisión del documento especificado, este programa utiliza una API del lado del cliente para verificar esa revisión.

3. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/GetRevision.js
```

Si ya no necesita usar el libro mayor `vehicle-registration`, continúe con Paso 8 (opcional): limpiar recursos.

Paso 8 (opcional): limpiar recursos

Puede seguir utilizando el libro mayor `vehicle-registration`. Sin embargo, si ya no lo necesita, debe eliminarlo.

Para eliminar el libro mayor

1. Utilice el siguiente programa (`DeleteLedger.ts`) para eliminar el libro mayor `vehicle-registration` y todo su contenido.

```
/*
 * Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: MIT-0
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
 * software and associated documentation files (the "Software"), to deal in the
 Software
 * without restriction, including without limitation the rights to use, copy,
 modify,
 * merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
 * permit persons to whom the Software is furnished to do so.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
 * INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
 * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
 * HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
 * 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**
>
> Si la protección contra eliminación está habilitada para su libro mayor, primero debe desactivarla para poder eliminar el libro mayor utilizando la API de QLDB.

2. Para ejecutar el programa transpilado, introduzca el siguiente comando.

```
node dist/DeleteLedger.js
```

Tutorial de Python para Amazon QLDB

En esta implementación de la aplicación de ejemplo del tutorial, utilizará el controlador Amazon QLDB con AWS SDK for Python (Boto3) para crear un libro mayor de QLDB y rellenarlo con datos de ejemplo.

Mientras sigue este tutorial, puede consultar el cliente de nivel bajo de QLDB en la referencia de API SDK de AWS para Python (Boto3) para consultar las operaciones de API de administración. Para las operaciones de datos transaccionales, puede consultar el controlador QLDB para la referencia de la API de Python.

> **Note**
>
> Cuando proceda, algunos pasos del tutorial incluyen comandos o ejemplos de código diferentes para cada versión principal compatible del controlador QLDB para Python.

Temas

- Paso 8 (opcional): limpiar recursos

Instalación de la aplicación de ejemplo Python de Amazon QLDB

En esta sección se describe cómo instalar y ejecutar la aplicación de ejemplo de Amazon QLDB proporcionada para este tutorial de Python paso a paso. El caso de uso de esta aplicación de ejemplo es una base de datos del Departamento de Vehículos Automóviles (DMV) que rastrea la información histórica completa de las matriculaciones de vehículos.

Esta aplicación de ejemplo de DMV para Python es de código abierto y se encuentra en el repositorio de GitHub aws-samples/amazon-qldb-dmv-sample-python.

Requisitos previos

Antes de comenzar, asegúrese de haber completado la configuración del controlador QLDB para Python Requisitos previos. Esto incluye instalar Python y hacer lo siguiente:

1. Regístrese en AWS.
2. Cree un usuario con los permisos de QLDB adecuados.
3. Conceda acceso programático de desarrollo.

Para completar todos los pasos de este tutorial, necesitará acceso administrativo completo a su recurso de libro mayor a través de la API de QLDB.

Instalación

Para instalar la aplicación de muestra

1. Ejecute el siguiente comando `pip` para clonar la aplicación de muestra desde GitHub.

   3.x

   ```
   pip install git+https://github.com/aws-samples/amazon-qldb-dmv-sample-python.git
   ```

   2.x

   ```
   pip install git+https://github.com/aws-samples/amazon-qldb-dmv-sample-
   python.git@v1.0.0
   ```

La aplicación de ejemplo empaqueta el código origen completo de este tutorial y sus dependencias, incluidos el controlador Python y AWS SDK for Python (Boto3).

2. Antes de empezar a ejecutar el código en la línea de comandos, cambie su directorio de trabajo actual a la ubicación en la que está instalado el paquete `pyqldbsamples`. Ingrese el siguiente comando.

```
cd $(python -c "import pyqldbsamples; print(pyqldbsamples.__path__[0])")
```

3. Continúe con Paso 1: crear un nuevo libro mayor para iniciar el tutorial y crear un libro mayor.

Paso 1: crear un nuevo libro mayor

En este paso, creará un nuevo libro de mayor de Amazon QLDB denominado `vehicle-registration`.

Para crear un nuevo libro mayor

1. Revise el siguiente archivo (`constants.py`), que contiene valores constantes que utilizan todos los demás programas de este tutorial.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


class Constants:
    """
    Constant values used throughout this tutorial.
    """
    LEDGER_NAME = "vehicle-registration"

    VEHICLE_REGISTRATION_TABLE_NAME = "VehicleRegistration"
    VEHICLE_TABLE_NAME = "Vehicle"
    PERSON_TABLE_NAME = "Person"
    DRIVERS_LICENSE_TABLE_NAME = "DriversLicense"

    LICENSE_NUMBER_INDEX_NAME = "LicenseNumber"
    GOV_ID_INDEX_NAME = "GovId"
    VEHICLE_VIN_INDEX_NAME = "VIN"
    LICENSE_PLATE_NUMBER_INDEX_NAME = "LicensePlateNumber"
    PERSON_ID_INDEX_NAME = "PersonId"

    JOURNAL_EXPORT_S3_BUCKET_NAME_PREFIX = "qldb-tutorial-journal-export"
    USER_TABLES = "information_schema.user_tables"
    S3_BUCKET_ARN_TEMPLATE = "arn:aws:s3:::"
    LEDGER_NAME_WITH_TAGS = "tags"

    RETRY_LIMIT = 4
```

2. Utilice el siguiente programa (`create_ledger.py`) para crear un libro mayor denominado `vehicle-registration`.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
```

```
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
from time import sleep

from boto3 import client

from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')

LEDGER_CREATION_POLL_PERIOD_SEC = 10
ACTIVE_STATE = "ACTIVE"


def create_ledger(name):
    """
    Create a new ledger with the specified name.

    :type name: str
    :param name: Name for the ledger to be created.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info("Let's create the ledger named: {}...".format(name))
    result = qldb_client.create_ledger(Name=name, PermissionsMode='ALLOW_ALL')
    logger.info('Success. Ledger state: {}.'.format(result.get('State')))
    return result


def wait_for_active(name):
```

```
    """
    Wait for the newly created ledger to become active.

    :type name: str
    :param name: The ledger to check on.

    :rtype: dict
    :return: Result from the request.
    """
    logger.info('Waiting for ledger to become active...')
    while True:
        result = qldb_client.describe_ledger(Name=name)
        if result.get('State') == ACTIVE_STATE:
            logger.info('Success. Ledger is active and ready to use.')
            return result
        logger.info('The ledger is still creating. Please wait...')
        sleep(LEDGER_CREATION_POLL_PERIOD_SEC)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create a ledger and wait for it to be active.
    """
    try:
        create_ledger(ledger_name)
        wait_for_active(ledger_name)
    except Exception as e:
        logger.exception('Unable to create the ledger!')
        raise e


if __name__ == '__main__':
    main()
```

> **Note**
>
> - En la llamada `create_ledger`, debe especificar un nombre de libro mayor y un modo de permisos. Recomendamos encarecidamente el modo de permisos `STANDARD` para maximizar la seguridad de los datos del libro mayor.
> - Al crear un libro mayor, se habilita de forma predeterminada la protección contra la eliminación. Se trata de una característica de QLDB que impide que los libros mayores

sean eliminados por cualquier usuario. Tiene la opción de deshabilitar la protección contra la eliminación al crear el libro mayor con la API QLDB o AWS Command Line Interface (AWS CLI).

- Si lo desea, también puede especificar etiquetas para adjuntar al libro mayor.

3. Para ejecutar el programa, introduzca el siguiente comando.

```
python create_ledger.py
```

Para comprobar la conexión con el nuevo libro mayor, continúe con Paso 2: probar la conectividad con el libro mayor.

Paso 2: probar la conectividad con el libro mayor

En este paso, verifica que puede conectarse al libro mayor `vehicle-registration` de Amazon QLDB mediante el punto de conexión de la API de datos transaccionales.

Para probar la conectividad con el libro mayor

1. Utilice el siguiente programa (`connect_to_ledger.py`) para crear una conexión de sesión de datos al libro mayor `vehicle-registration`.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

```
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from botocore.exceptions import ClientError

from pyqldb.driver.qldb_driver import QldbDriver
from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_qldb_driver(ledger_name=Constants.LEDGER_NAME, region_name=None,
 endpoint_url=None, boto3_session=None):
    """
    Create a QLDB driver for executing transactions.

    :type ledger_name: str
    :param ledger_name: The QLDB ledger name.

    :type region_name: str
    :param region_name: See [1].

    :type endpoint_url: str
    :param endpoint_url: See [1].

    :type boto3_session: :py:class:`boto3.session.Session`
    :param boto3_session: The boto3 session to create the client with (see [1]).

    :rtype: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :return: A QLDB driver object.

    [1]: `Boto3 Session.client Reference <https://
boto3.amazonaws.com/v1/documentation/api/latest/reference/core/
session.html#boto3.session.Session.client>`.
    """
```

```
    qldb_driver = QldbDriver(ledger_name=ledger_name, region_name=region_name,
 endpoint_url=endpoint_url,
                             boto3_session=boto3_session)
    return qldb_driver


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Connect to a given ledger using default settings.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            logger.info('Listing table names ')
            for table in driver.list_tables():
                logger.info(table)
    except ClientError as ce:
        logger.exception('Unable to list tables.')
        raise ce


if __name__ == '__main__':
    main()
```

> **Note**
>
> - Para ejecutar transacciones de datos en su libro mayor, debe crear un objeto de controlador de QLDB para conectarse a un libro mayor específico. Se trata de un objeto de cliente diferente al objeto `qldb_client` que utilizó en el paso anterior para crear el libro mayor. Ese cliente anterior solo se usa para ejecutar las operaciones de la API de administración que se enumeran en Referencia de la API de Amazon QLDB.
> - Debe especificar un nombre de libro mayor al crear este objeto de controlador.

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
```

```
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from botocore.exceptions import ClientError

from pyqldb.driver.pooled_qldb_driver import PooledQldbDriver
from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_qldb_driver(ledger_name=Constants.LEDGER_NAME, region_name=None,
 endpoint_url=None, boto3_session=None):
    """
    Create a QLDB driver for creating sessions.

    :type ledger_name: str
    :param ledger_name: The QLDB ledger name.

    :type region_name: str
    :param region_name: See [1].
```

```
    :type endpoint_url: str
    :param endpoint_url: See [1].

    :type boto3_session: :py:class:`boto3.session.Session`
    :param boto3_session: The boto3 session to create the client with (see [1]).

    :rtype: :py:class:`pyqldb.driver.pooled_qldb_driver.PooledQldbDriver`
    :return: A pooled QLDB driver object.

    [1]: `Boto3 Session.client Reference <https://
boto3.amazonaws.com/v1/documentation/api/latest/reference/core/
session.html#boto3.session.Session.client>`.
    """
    qldb_driver = PooledQldbDriver(ledger_name=ledger_name,
 region_name=region_name, endpoint_url=endpoint_url,
                                   boto3_session=boto3_session)
    return qldb_driver


def create_qldb_session():
    """
    Retrieve a QLDB session object.

    :rtype: :py:class:`pyqldb.session.pooled_qldb_session.PooledQldbSession`
    :return: A pooled QLDB session object.
    """
    qldb_session = pooled_qldb_driver.get_session()
    return qldb_session


pooled_qldb_driver = create_qldb_driver()


if __name__ == '__main__':
    """
    Connect to a session for a given ledger using default settings.
    """
    try:
        qldb_session = create_qldb_session()
        logger.info('Listing table names ')
        for table in qldb_session.list_tables():
            logger.info(table)
    except ClientError:
```

```
        logger.exception('Unable to create session.')
```

> **Note**
>
> - Para ejecutar transacciones de datos en su libro mayor, debe crear un objeto de controlador de QLDB para conectarse a un libro mayor específico. Se trata de un objeto de cliente diferente al objeto `qldb_client` que utilizó en el paso anterior para crear el libro mayor. Ese cliente anterior solo se usa para ejecutar las operaciones de la API de administración que se enumeran en Referencia de la API de Amazon QLDB.
> - Primero, cree un objeto controlador de QLDB agrupado. Debe especificar un nombre de libro mayor al crear este controlador.
> - A continuación, puede crear sesiones a partir de este objeto de controlador agrupado.

2. Para ejecutar el programa, introduzca el siguiente comando.

```
python connect_to_ledger.py
```

Para crear tablas en el libro mayor `vehicle-registration`, continúe con Paso 3: cree tablas, índices y datos de muestra.

Paso 3: cree tablas, índices y datos de muestra

Cuando su libro mayor de Amazon QLDB esté activo y acepte conexiones, podrá empezar a crear tablas con datos sobre los vehículos, sus propietarios y su información de registro. Tras crear las tablas y los índices, puede cargarlos con datos.

En este paso, creará cuatro tablas en el libro mayor `vehicle-registration`:

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

También se crean los siguientes índices.

Nombre de la tabla	Campo
`VehicleRegistration`	`VIN`
`VehicleRegistration`	`LicensePlateNumber`
`Vehicle`	`VIN`
`Person`	`GovId`
`DriversLicense`	`LicenseNumber`
`DriversLicense`	`PersonId`

Al insertar datos de ejemplo, primero debe insertar los documentos en la tabla `Person`. A continuación, utilice los `id` asignados por el sistema de cada documento `Person` para rellenar los campos pertinentes en los documentos `VehicleRegistration` y `DriversLicense` correspondientes.

> **Tip**
>
> Como práctica recomendada, utilice un `id` de documento asignado por el sistema como clave externa. Si bien puede definir campos que pretenden ser identificadores únicos (por ejemplo, el número de chasis [VIN] de un vehículo), el verdadero identificador único de un documento es su `id`. Este campo se incluye en los metadatos del documento, que puede consultar en la vista confirmada (la vista de una tabla definida por el sistema).
> Para obtener más información acerca de las vistas en QLDB, consulte Conceptos clave.
> Para obtener más información sobre metadatos, consulte Consulta de los metadatos del documento.

Para crear tablas e índices

1. Utilice el siguiente programa (`create_table.py`) para crear las tablas mencionadas anteriormente.

3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_table(driver, table_name):
    """
    Create a table with the specified name.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
```

```
    :type table_name: str
    :param table_name: Name of the table to create.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating the '{}' table...".format(table_name))
    statement = 'CREATE TABLE {}'.format(table_name)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement))
    logger.info('{} table created successfully.'.format(table_name))
    return len(list(cursor))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create registrations, vehicles, owners, and licenses tables.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
            create_table(driver, Constants.DRIVERS_LICENSE_TABLE_NAME)
            create_table(driver, Constants.PERSON_TABLE_NAME)
            create_table(driver, Constants.VEHICLE_TABLE_NAME)
            create_table(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME)
            logger.info('Tables created successfully.')
    except Exception as e:
        logger.exception('Errors creating tables.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
```

```
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_table(transaction_executor, table_name):
    """
    Create a table with the specified name using an Executor object.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to create.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating the '{}' table...".format(table_name))
    statement = 'CREATE TABLE {}'.format(table_name)
    cursor = transaction_executor.execute_statement(statement)
```

```
    logger.info('{} table created successfully.'.format(table_name))
    return len(list(cursor))


if __name__ == '__main__':
    """
    Create registrations, vehicles, owners, and licenses tables in a single
 transaction.
    """
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda x: create_table(x,
 Constants.DRIVERS_LICENSE_TABLE_NAME) and
                                   create_table(x, Constants.PERSON_TABLE_NAME)
 and
                                   create_table(x, Constants.VEHICLE_TABLE_NAME)
 and
                                   create_table(x,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Tables created successfully.')
    except Exception:
        logger.exception('Errors creating tables.')
```

> **Note**
>
> Este programa muestra cómo utilizar la función `execute_lambda`. En este ejemplo, ejecuta varias instrucciones `CREATE TABLE` de PartiQL con una sola expresión lambda. Esta función de ejecución inicia implícitamente una transacción, ejecuta todas las instrucciones de lambda y, a continuación, confirma automáticamente la transacción.

2. Para ejecutar el programa, introduzca el siguiente comando.

```
python create_table.py
```

3. Utilice el siguiente programa (`create_index.py`) para crear índices en las tablas, tal y como se describió anteriormente.

3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_index(driver, table_name, index_attribute):
    """
    Create an index for a particular table.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
```

```
    :type table_name: str
    :param table_name: Name of the table to add indexes for.

    :type index_attribute: str
    :param index_attribute: Index to create on a single attribute.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating index on '{}'...".format(index_attribute))
    statement = 'CREATE INDEX on {} ({})'.format(table_name, index_attribute)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement))
    return len(list(cursor))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Create indexes on tables in a particular ledger.
    """
    logger.info('Creating indexes on all tables...')
    try:
        with create_qldb_driver(ledger_name) as driver:
            create_index(driver, Constants.PERSON_TABLE_NAME,
 Constants.GOV_ID_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_TABLE_NAME,
 Constants.VEHICLE_VIN_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.LICENSE_PLATE_NUMBER_INDEX_NAME)
            create_index(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 Constants.VEHICLE_VIN_INDEX_NAME)
            create_index(driver, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.PERSON_ID_INDEX_NAME)
            create_index(driver, Constants.DRIVERS_LICENSE_TABLE_NAME,
 Constants.LICENSE_NUMBER_INDEX_NAME)
            logger.info('Indexes created successfully.')
    except Exception as e:
        logger.exception('Unable to create indexes.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def create_index(transaction_executor, table_name, index_attribute):
    """
    Create an index for a particular table.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
```

```
    :type table_name: str
    :param table_name: Name of the table to add indexes for.

    :type index_attribute: str
    :param index_attribute: Index to create on a single attribute.

    :rtype: int
    :return: The number of changes to the database.
    """
    logger.info("Creating index on '{}'...".format(index_attribute))
    statement = 'CREATE INDEX on {} ({})'.format(table_name, index_attribute)
    cursor = transaction_executor.execute_statement(statement)
    return len(list(cursor))


if __name__ == '__main__':
    """
    Create indexes on tables in a particular ledger.
    """
    logger.info('Creating indexes on all tables in a single transaction...')
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda x: create_index(x,
 Constants.PERSON_TABLE_NAME,

 Constants.GOV_ID_INDEX_NAME)
                                   and create_index(x,
 Constants.VEHICLE_TABLE_NAME,

 Constants.VEHICLE_VIN_INDEX_NAME)
                                   and create_index(x,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.LICENSE_PLATE_NUMBER_INDEX_NAME)
                                   and create_index(x,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME,

 Constants.VEHICLE_VIN_INDEX_NAME)
                                   and create_index(x,
 Constants.DRIVERS_LICENSE_TABLE_NAME,

 Constants.PERSON_ID_INDEX_NAME)
```

```
                                    and create_index(x,
Constants.DRIVERS_LICENSE_TABLE_NAME,

Constants.LICENSE_NUMBER_INDEX_NAME),
                                    lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
            logger.info('Indexes created successfully.')
    except Exception:
        logger.exception('Unable to create indexes.')
```

4. Para ejecutar el programa, introduzca el siguiente comando.

```
python create_index.py
```

Para cargar los datos en las tablas

1. Revise el siguiente archivo (`sample_data.py`), que representa los datos de ejemplo que inserta en las tablas `vehicle-registration`. Este archivo también se importa del paquete `amazon.ion` para proporcionar funciones auxiliares que convierten, analizan e imprimen datos de Amazon Ion.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
```

```
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
from datetime import datetime
from decimal import Decimal
from logging import basicConfig, getLogger, INFO

from amazon.ion.simple_types import IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict,
 IonPyFloat, IonPyInt, IonPyList, \
    IonPyNull, IonPySymbol, IonPyText, IonPyTimestamp
from amazon.ion.simpleion import dumps, loads

logger = getLogger(__name__)
basicConfig(level=INFO)
IonValue = (IonPyBool, IonPyBytes, IonPyDecimal, IonPyDict, IonPyFloat, IonPyInt,
 IonPyList, IonPyNull, IonPySymbol,
            IonPyText, IonPyTimestamp)


class SampleData:
    """
    Sample domain objects for use throughout this tutorial.
    """
    DRIVERS_LICENSE = [
        {
            'PersonId': '',
            'LicenseNumber': 'LEWISR261LL',
            'LicenseType': 'Learner',
            'ValidFromDate': datetime(2016, 12, 20),
            'ValidToDate': datetime(2020, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'LOGANB486CG',
            'LicenseType': 'Probationary',
            'ValidFromDate': datetime(2016, 4, 6),
            'ValidToDate': datetime(2020, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': '744 849 301',
            'LicenseType': 'Full',
            'ValidFromDate': datetime(2017, 12, 6),
            'ValidToDate': datetime(2022, 10, 15)
        },
        {
```

```
            'PersonId': '',
            'LicenseNumber': 'P626-168-229-765',
            'LicenseType': 'Learner',
            'ValidFromDate': datetime(2017, 8, 16),
            'ValidToDate': datetime(2021, 11, 15)
        },
        {
            'PersonId': '',
            'LicenseNumber': 'S152-780-97-415-0',
            'LicenseType': 'Probationary',
            'ValidFromDate': datetime(2015, 8, 15),
            'ValidToDate': datetime(2021, 8, 21)
        }
    ]
    PERSON = [
        {
            'FirstName': 'Raul',
            'LastName': 'Lewis',
            'Address': '1719 University Street, Seattle, WA, 98109',
            'DOB': datetime(1963, 8, 19),
            'GovId': 'LEWISR261LL',
            'GovIdType': 'Driver License'
        },
        {
            'FirstName': 'Brent',
            'LastName': 'Logan',
            'DOB': datetime(1967, 7, 3),
            'Address': '43 Stockert Hollow Road, Everett, WA, 98203',
            'GovId': 'LOGANB486CG',
            'GovIdType': 'Driver License'
        },
        {
            'FirstName': 'Alexis',
            'LastName': 'Pena',
            'DOB': datetime(1974, 2, 10),
            'Address': '4058 Melrose Street, Spokane Valley, WA, 99206',
            'GovId': '744 849 301',
            'GovIdType': 'SSN'
        },
        {
            'FirstName': 'Melvin',
            'LastName': 'Parker',
            'DOB': datetime(1976, 5, 22),
            'Address': '4362 Ryder Avenue, Seattle, WA, 98101',
```

```
            'GovId': 'P626-168-229-765',
            'GovIdType': 'Passport'
        },
        {
            'FirstName': 'Salvatore',
            'LastName': 'Spencer',
            'DOB': datetime(1997, 11, 15),
            'Address': '4450 Honeysuckle Lane, Seattle, WA, 98101',
            'GovId': 'S152-780-97-415-0',
            'GovIdType': 'Passport'
        }
    ]
    VEHICLE = [
        {
            'VIN': '1N4AL11D75C109151',
            'Type': 'Sedan',
            'Year': 2011,
            'Make': 'Audi',
            'Model': 'A5',
            'Color': 'Silver'
        },
        {
            'VIN': 'KM8SRDHF6EU074761',
            'Type': 'Sedan',
            'Year': 2015,
            'Make': 'Tesla',
            'Model': 'Model S',
            'Color': 'Blue'
        },
        {
            'VIN': '3HGGK5G53FM761765',
            'Type': 'Motorcycle',
            'Year': 2011,
            'Make': 'Ducati',
            'Model': 'Monster 1200',
            'Color': 'Yellow'
        },
        {
            'VIN': '1HVBBAANXWH544237',
            'Type': 'Semi',
            'Year': 2009,
            'Make': 'Ford',
            'Model': 'F 150',
            'Color': 'Black'
```

```
        },
        {
            'VIN': '1C4RJFAG0FC625797',
            'Type': 'Sedan',
            'Year': 2019,
            'Make': 'Mercedes',
            'Model': 'CLK 350',
            'Color': 'White'
        }
    ]
    VEHICLE_REGISTRATION = [
        {
            'VIN': '1N4AL11D75C109151',
            'LicensePlateNumber': 'LEWISR261LL',
            'State': 'WA',
            'City': 'Seattle',
            'ValidFromDate': datetime(2017, 8, 21),
            'ValidToDate': datetime(2020, 5, 11),
            'PendingPenaltyTicketAmount': Decimal('90.25'),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': 'KM8SRDHF6EU074761',
            'LicensePlateNumber': 'CA762X',
            'State': 'WA',
            'City': 'Kent',
            'PendingPenaltyTicketAmount': Decimal('130.75'),
            'ValidFromDate': datetime(2017, 9, 14),
            'ValidToDate': datetime(2020, 6, 25),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '3HGGK5G53FM761765',
            'LicensePlateNumber': 'CD820Z',
            'State': 'WA',
            'City': 'Everett',
            'PendingPenaltyTicketAmount': Decimal('442.30'),
            'ValidFromDate': datetime(2011, 3, 17),
```

```
            'ValidToDate': datetime(2021, 3, 24),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '1HVBBAANXWH544237',
            'LicensePlateNumber': 'LS477D',
            'State': 'WA',
            'City': 'Tacoma',
            'PendingPenaltyTicketAmount': Decimal('42.20'),
            'ValidFromDate': datetime(2011, 10, 26),
            'ValidToDate': datetime(2023, 9, 25),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        },
        {
            'VIN': '1C4RJFAG0FC625797',
            'LicensePlateNumber': 'TH393F',
            'State': 'WA',
            'City': 'Olympia',
            'PendingPenaltyTicketAmount': Decimal('30.45'),
            'ValidFromDate': datetime(2013, 9, 2),
            'ValidToDate': datetime(2024, 3, 19),
            'Owners': {
                'PrimaryOwner': {'PersonId': ''},
                'SecondaryOwners': []
            }
        }
    ]


def convert_object_to_ion(py_object):
    """
    Convert a Python object into an Ion object.

    :type py_object: object
    :param py_object: The object to convert.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyValue`
    :return: The converted Ion object.
```

```
    """
    ion_object = loads(dumps(py_object))
    return ion_object


def to_ion_struct(key, value):
    """
    Convert the given key and value into an Ion struct.

    :type key: str
    :param key: The key which serves as an unique identifier.

    :type value: str
    :param value: The value associated with a given key.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The Ion dictionary object.
    """
    ion_struct = dict()
    ion_struct[key] = value
    return loads(str(ion_struct))


def get_document_ids(transaction_executor, table_name, field, value):
    """
    Gets the document IDs from the given table.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: The table name to query.

    :type field: str
    :param field: A field to query.

    :type value: str
    :param value: The key of the given field.

    :rtype: list
    :return: A list of document IDs.
    """
    query = "SELECT id FROM {} AS t BY id WHERE t.{} = ?".format(table_name, field)
```

```
    cursor = transaction_executor.execute_statement(query,
 convert_object_to_ion(value))
    return list(map(lambda table: table.get('id'), cursor))


def get_document_ids_from_dml_results(result):
    """
    Return a list of modified document IDs as strings from DML results.

    :type result: :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :param: result: The result set from DML operation.

    :rtype: list
    :return: List of document IDs.
    """
    ret_val = list(map(lambda x: x.get('documentId'), result))
    return ret_val


def print_result(cursor):
    """
    Pretty print the result set. Returns the number of documents in the result set.

    :type cursor: :py:class:`pyqldb.cursor.stream_cursor.StreamCursor`/
                  :py:class:`pyqldb.cursor.buffered_cursor.BufferedCursor`
    :param cursor: An instance of the StreamCursor or BufferedCursor class.

    :rtype: int
    :return: Number of documents in the result set.
    """
    result_counter = 0
    for row in cursor:
        # Each row would be in Ion format.
        print_ion(row)
        result_counter += 1
    return result_counter


def print_ion(ion_value):
    """
    Pretty print an Ion Value.

    :type ion_value: :py:class:`amazon.ion.simple_types.IonPySymbol`
    :param ion_value: Any Ion Value to be pretty printed.
```

```
    """
    logger.info(dumps(ion_value, binary=False, indent='  ',
omit_version_marker=True))
```

> **Note**
>
> La función `get_document_ids` ejecuta una consulta que devuelve los identificadores de documentos asignados por el sistema desde una tabla. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

2. Utilice el siguiente programa (`insert_document.py`) para insertar los datos de ejemplo en las tablas.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
```

```
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion, SampleData,
 get_document_ids_from_dml_results
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def update_person_id(document_ids):
    """
    Update the PersonId value for DriversLicense records and the PrimaryOwner
 value for VehicleRegistration records.

    :type document_ids: list
    :param document_ids: List of document IDs.

    :rtype: list
    :return: Lists of updated DriversLicense records and updated
 VehicleRegistration records.
    """
    new_drivers_licenses = SampleData.DRIVERS_LICENSE.copy()
    new_vehicle_registrations = SampleData.VEHICLE_REGISTRATION.copy()
    for i in range(len(SampleData.PERSON)):
        drivers_license = new_drivers_licenses[i]
        registration = new_vehicle_registrations[i]
        drivers_license.update({'PersonId': str(document_ids[i])})
        registration['Owners']['PrimaryOwner'].update({'PersonId':
 str(document_ids[i])})
    return new_drivers_licenses, new_vehicle_registrations


def insert_documents(driver, table_name, documents):
    """
    Insert the given list of documents into a table in a single transaction.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type table_name: str
    :param table_name: Name of the table to insert documents into.

    :type documents: list
```

```
    :param documents: List of documents to insert.

    :rtype: list
    :return: List of documents IDs for the newly inserted documents.
    """
    logger.info('Inserting some documents in the {}
 table...'.format(table_name))
    statement = 'INSERT INTO {} ?'.format(table_name)
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement,

 convert_object_to_ion(documents)))
    list_of_document_ids = get_document_ids_from_dml_results(cursor)

    return list_of_document_ids


def update_and_insert_documents(driver):
    """
    Handle the insertion of documents and updating PersonIds.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
    """
    list_ids = insert_documents(driver, Constants.PERSON_TABLE_NAME,
 SampleData.PERSON)

    logger.info("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...")
    new_licenses, new_registrations = update_person_id(list_ids)

    insert_documents(driver, Constants.VEHICLE_TABLE_NAME, SampleData.VEHICLE)
    insert_documents(driver, Constants.VEHICLE_REGISTRATION_TABLE_NAME,
 new_registrations)
    insert_documents(driver, Constants.DRIVERS_LICENSE_TABLE_NAME, new_licenses)


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Insert documents into a table in a QLDB ledger.
    """
    try:
        with create_qldb_driver(ledger_name) as driver:
```

```
            # An INSERT statement creates the initial revision of a document
 with a version number of zero.
            # QLDB also assigns a unique document identifier in GUID format as
 part of the metadata.
            update_and_insert_documents(driver)
            logger.info('Documents inserted successfully!')
    except Exception as e:
        logger.exception('Error inserting or updating documents.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO
```

```
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion, SampleData,
 get_document_ids_from_dml_results
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def update_person_id(document_ids):
    """
    Update the PersonId value for DriversLicense records and the PrimaryOwner
 value for VehicleRegistration records.

    :type document_ids: list
    :param document_ids: List of document IDs.

    :rtype: list
    :return: Lists of updated DriversLicense records and updated
 VehicleRegistration records.
    """
    new_drivers_licenses = SampleData.DRIVERS_LICENSE.copy()
    new_vehicle_registrations = SampleData.VEHICLE_REGISTRATION.copy()
    for i in range(len(SampleData.PERSON)):
        drivers_license = new_drivers_licenses[i]
        registration = new_vehicle_registrations[i]
        drivers_license.update({'PersonId': str(document_ids[i])})
        registration['Owners']['PrimaryOwner'].update({'PersonId':
 str(document_ids[i])})
    return new_drivers_licenses, new_vehicle_registrations


def insert_documents(transaction_executor, table_name, documents):
    """
    Insert the given list of documents into a table in a single transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type table_name: str
    :param table_name: Name of the table to insert documents into.

    :type documents: list
```

```
    :param documents: List of documents to insert.

    :rtype: list
    :return: List of documents IDs for the newly inserted documents.
    """
    logger.info('Inserting some documents in the {}
 table...'.format(table_name))
    statement = 'INSERT INTO {} ?'.format(table_name)
    cursor = transaction_executor.execute_statement(statement,
 convert_object_to_ion(documents))
    list_of_document_ids = get_document_ids_from_dml_results(cursor)

    return list_of_document_ids


def update_and_insert_documents(transaction_executor):
    """
    Handle the insertion of documents and updating PersonIds all in a single
 transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    """
    list_ids = insert_documents(transaction_executor,
 Constants.PERSON_TABLE_NAME, SampleData.PERSON)

    logger.info("Updating PersonIds for 'DriversLicense' and PrimaryOwner for
 'VehicleRegistration'...")
    new_licenses, new_registrations = update_person_id(list_ids)

    insert_documents(transaction_executor, Constants.VEHICLE_TABLE_NAME,
 SampleData.VEHICLE)
    insert_documents(transaction_executor,
 Constants.VEHICLE_REGISTRATION_TABLE_NAME, new_registrations)
    insert_documents(transaction_executor, Constants.DRIVERS_LICENSE_TABLE_NAME,
 new_licenses)


if __name__ == '__main__':
    """
    Insert documents into a table in a QLDB ledger.
    """
    try:
```

```
        with create_qldb_session() as session:
            # An INSERT statement creates the initial revision of a document
with a version number of zero.
            # QLDB also assigns a unique document identifier in GUID format as
part of the metadata.
            session.execute_lambda(lambda executor:
update_and_insert_documents(executor),
                                   lambda retry_attempt: logger.info('Retrying
due to OCC conflict...'))
            logger.info('Documents inserted successfully!')
    except Exception:
        logger.exception('Error inserting or updating documents.')
```

> **Note**
>
> - Este programa muestra cómo llamar a la función `execute_statement` con valores parametrizados. Puede pasar parámetros de datos además de la instrucción PartiQL que desee ejecutar. Utilice un signo de interrogación (?) como marcador de posición variable en la cadena de la instrucción.
> - Si una instrucción `INSERT` tiene éxito, devuelve `id` de cada documento insertado.

3. Para ejecutar el programa, introduzca el siguiente comando.

```
python insert_document.py
```

A continuación, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`. Continúe en Paso 4: consultar las tablas en un libro mayor.

Paso 4: consultar las tablas en un libro mayor

Tras crear tablas en un libro mayor de Amazon QLDB y cargarlas con datos, puede ejecutar consultas para revisar los datos de registro del vehículo que acaba de insertar. QLDB emplea PartiQL como lenguaje de consulta y Amazon Ion como modelo de datos orientado a documentos.

PartiQL es un lenguaje de consulta de código abierto compatible con SQL que se ha ampliado para funcionar con Ion. PartiQL le permite insertar, consultar y administrar sus datos con operadores SQL conocidos. Amazon Ion es un superconjunto de JSON. Ion es un formato de datos de código abierto

basado en documentos que le brinda la flexibilidad de almacenar y procesar datos estructurados, semiestructurados y anidados.

En este paso, puede usar las instrucciones `SELECT` para leer los datos de las tablas del libro mayor `vehicle-registration`.

> **Warning**
>
> Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. PartiQL admite este tipo de consultas porque es compatible con SQL. Sin embargo, no ejecute escaneados de tablas para casos de uso de producción en QLDB. Los escaneados de tablas pueden provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad en un campo indexado o en un ID de documento, por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Para consultar las tablas

1. Utilice el siguiente programa (`find_vehicles.py`) para consultar todos los vehículos registrados a nombre de una persona en su libro mayor.

   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**
>
> En primer lugar, este programa consulta la tabla `Person` del documento con `GovId` `LEWISR261LL` para obtener su campo de metadatos `id`.
> A continuación, utiliza este `id` de documento como clave externa para consultar la tabla `VehicleRegistration` mediante `PrimaryOwner.PersonId`. También combina `VehicleRegistration` con la tabla `Vehicle` del campo `VIN`.

2. Para ejecutar el programa, introduzca el siguiente comando.

```
python find_vehicles.py
```

Para obtener información sobre la modificación de los documentos en las tablas del libro mayor `vehicle-registration`, consulte Paso 5: modificar los documentos de un libro mayor.

Paso 5: modificar los documentos de un libro mayor

Ahora que tiene datos con los que trabajar, puede empezar a realizar cambios en los documentos del libro mayor `vehicle-registration` de Amazon QLDB. En este paso, los siguientes ejemplos de código muestran cómo ejecutar instrucciones de lenguaje de manipulación de datos (DML). Estas instrucciones actualizan al propietario principal de un vehículo y añaden un propietario secundario a otro vehículo.

Para modificar documentos

1. Use el siguiente programa (`transfer_vehicle_ownership.py`) para actualizar el propietario principal del vehículo con VIN `1N4AL11D75C109151` en su libro mayor.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.add_secondary_owner import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)
```

```
def find_person_from_document_id(transaction_executor, document_id):
    """
    Query a driver's information using the given ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: The document ID required to query for the person.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    query = 'SELECT p.* FROM Person AS p BY pid WHERE pid = ?'
    cursor = transaction_executor.execute_statement(query, document_id)
    return next(cursor)


def find_primary_owner_for_vehicle(driver, vin):
    """
    Find the primary owner of a vehicle given its VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN to find primary owner for.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    logger.info('Finding primary owner for vehicle with VIN: {}.'.format(vin))
    query = "SELECT Owners.PrimaryOwner.PersonId FROM VehicleRegistration AS v
 WHERE v.VIN = ?"
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, convert_object_to_ion(vin)))
    try:
        return driver.execute_lambda(lambda executor:
 find_person_from_document_id(executor,

   next(cursor).get('PersonId')))
    except StopIteration:
```

```
        logger.error('No primary owner registered for this vehicle.')
        return None


def update_vehicle_registration(driver, vin, document_id):
    """
    Update the primary owner for a vehicle using the given VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: The VIN for the vehicle to operate on.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: New PersonId for the primary owner.

    :raises RuntimeError: If no vehicle registration was found using the given
 document ID and VIN.
    """
    logger.info('Updating the primary owner for vehicle with Vin:
 {}...'.format(vin))
    statement = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?"
    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement, document_id,

 convert_object_to_ion(vin)))
    try:
        print_result(cursor)
        logger.info('Successfully transferred vehicle with VIN: {} to new
 owner.'.format(vin))
    except StopIteration:
        raise RuntimeError('Unable to transfer vehicle, could not find
 registration.')


def validate_and_update_registration(driver, vin, current_owner, new_owner):
    """
    Validate the current owner of the given vehicle and transfer its ownership
 to a new owner.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
```

```
    :type vin: str
    :param vin: The VIN of the vehicle to transfer ownership of.

    :type current_owner: str
    :param current_owner: The GovId of the current owner of the vehicle.

    :type new_owner: str
    :param new_owner: The GovId of the new owner of the vehicle.

    :raises RuntimeError: If unable to verify primary owner.
    """
    primary_owner = find_primary_owner_for_vehicle(driver, vin)
    if primary_owner is None or primary_owner['GovId'] != current_owner:
        raise RuntimeError('Incorrect primary owner identified for vehicle,
 unable to transfer.')

    document_ids = driver.execute_lambda(lambda executor:
 get_document_ids(executor, Constants.PERSON_TABLE_NAME,

 'GovId', new_owner))
    update_vehicle_registration(driver, vin, document_ids[0])


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Find primary owner for a particular vehicle's VIN.
    Transfer to another primary owner for a particular vehicle's VIN.
    """
    vehicle_vin = SampleData.VEHICLE[0]['VIN']
    previous_owner = SampleData.PERSON[0]['GovId']
    new_owner = SampleData.PERSON[1]['GovId']

    try:
        with create_qldb_driver(ledger_name) as driver:
            validate_and_update_registration(driver, vehicle_vin,
 previous_owner, new_owner)
    except Exception as e:
        logger.exception('Error updating VehicleRegistration.')
        raise e


if __name__ == '__main__':
```

```
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.add_secondary_owner import get_document_ids, print_result,
 SampleData
from pyqldbsamples.constants import Constants
from pyqldbsamples.model.sample_data import convert_object_to_ion
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def find_person_from_document_id(transaction_executor, document_id):
    """
```

```
    Query a driver's information using the given ID.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: The document ID required to query for the person.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    query = 'SELECT p.* FROM Person AS p BY pid WHERE pid = ?'
    cursor = transaction_executor.execute_statement(query, document_id)
    return next(cursor)


def find_primary_owner_for_vehicle(transaction_executor, vin):
    """
    Find the primary owner of a vehicle given its VIN.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN to find primary owner for.

    :rtype: :py:class:`amazon.ion.simple_types.IonPyDict`
    :return: The resulting document from the query.
    """
    logger.info('Finding primary owner for vehicle with VIN: {}.'.format(vin))
    query = "SELECT Owners.PrimaryOwner.PersonId FROM VehicleRegistration AS v
 WHERE v.VIN = ?"
    cursor = transaction_executor.execute_statement(query,
 convert_object_to_ion(vin))
    try:
        return find_person_from_document_id(transaction_executor,
 next(cursor).get('PersonId'))
    except StopIteration:
        logger.error('No primary owner registered for this vehicle.')
        return None
```

```
def update_vehicle_registration(transaction_executor, vin, document_id):
    """
    Update the primary owner for a vehicle using the given VIN.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN for the vehicle to operate on.

    :type document_id: :py:class:`amazon.ion.simple_types.IonPyText`
    :param document_id: New PersonId for the primary owner.

    :raises RuntimeError: If no vehicle registration was found using the given
 document ID and VIN.
    """
    logger.info('Updating the primary owner for vehicle with Vin:
 {}...'.format(vin))
    statement = "UPDATE VehicleRegistration AS r SET
 r.Owners.PrimaryOwner.PersonId = ? WHERE r.VIN = ?"
    cursor = transaction_executor.execute_statement(statement, document_id,
 convert_object_to_ion(vin))
    try:
        print_result(cursor)
        logger.info('Successfully transferred vehicle with VIN: {} to new
 owner.'.format(vin))
    except StopIteration:
        raise RuntimeError('Unable to transfer vehicle, could not find
 registration.')


def validate_and_update_registration(transaction_executor, vin, current_owner,
 new_owner):
    """
    Validate the current owner of the given vehicle and transfer its ownership
 to a new owner in a single transaction.

    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.

    :type vin: str
    :param vin: The VIN of the vehicle to transfer ownership of.
```

```
    :type current_owner: str
    :param current_owner: The GovId of the current owner of the vehicle.

    :type new_owner: str
    :param new_owner: The GovId of the new owner of the vehicle.

    :raises RuntimeError: If unable to verify primary owner.
    """
    primary_owner = find_primary_owner_for_vehicle(transaction_executor, vin)
    if primary_owner is None or primary_owner['GovId'] != current_owner:
        raise RuntimeError('Incorrect primary owner identified for vehicle,
 unable to transfer.')

    document_id = next(get_document_ids(transaction_executor,
 Constants.PERSON_TABLE_NAME, 'GovId', new_owner))

    update_vehicle_registration(transaction_executor, vin, document_id)


if __name__ == '__main__':
    """
    Find primary owner for a particular vehicle's VIN.
    Transfer to another primary owner for a particular vehicle's VIN.
    """
    vehicle_vin = SampleData.VEHICLE[0]['VIN']
    previous_owner = SampleData.PERSON[0]['GovId']
    new_owner = SampleData.PERSON[1]['GovId']

    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda executor:
 validate_and_update_registration(executor, vehicle_vin,

     previous_owner, new_owner),
                                   retry_indicator=lambda retry_attempt:
 logger.info('Retrying due to OCC conflict...'))
    except Exception:
        logger.exception('Error updating VehicleRegistration.')
```

2. Para ejecutar el programa, introduzca el siguiente comando.

```
python transfer_vehicle_ownership.py
```

3. Utilice el siguiente programa (`add_secondary_owner.py`) para añadir un propietario secundario al vehículo con el VIN `KM8SRDHF6EU074761` en su libro mayor.

   3.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import to_ion_struct, get_document_ids,
 print_result, SampleData, \
    convert_object_to_ion
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_driver

logger = getLogger(__name__)
basicConfig(level=INFO)


def get_document_id_by_gov_id(driver, government_id):
```

```
    """
    Find a driver's person ID using the given government ID.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type government_id: str
    :param government_id: A driver's government ID.

    :rtype: list
    :return: A list of document IDs.
    """
    logger.info("Finding secondary owner's person ID using given government ID:
 {}.".format(government_id))
    return driver.execute_lambda(lambda executor: get_document_ids(executor,
 Constants.PERSON_TABLE_NAME, 'GovId',

 government_id))


def is_secondary_owner_for_vehicle(driver, vin, secondary_owner_id):
    """
    Check whether a secondary owner has already been registered for the given
 VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to query.

    :type secondary_owner_id: str
    :param secondary_owner_id: The secondary owner's person ID.

    :rtype: bool
    :return: If the driver has already been registered.
    """
    logger.info('Finding secondary owners for vehicle with VIN:
 {}...'.format(vin))
    query = 'SELECT Owners.SecondaryOwners FROM VehicleRegistration AS v WHERE
 v.VIN = ?'
    rows = driver.execute_lambda(lambda executor:
 executor.execute_statement(query, convert_object_to_ion(vin)))
```

```
    for row in rows:
        secondary_owners = row.get('SecondaryOwners')
        person_ids = map(lambda owner: owner.get('PersonId').text,
 secondary_owners)
        if secondary_owner_id in person_ids:
            return True
    return False


def add_secondary_owner_for_vin(driver, vin, parameter):
    """
    Add a secondary owner into `VehicleRegistration` table for a particular VIN.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.

    :type vin: str
    :param vin: VIN of the vehicle to add a secondary owner for.

    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to
 Ion for filling in parameters of the
                      statement.
    """
    logger.info('Inserting secondary owner for vehicle with VIN:
 {}...'.format(vin))
    statement = "FROM VehicleRegistration AS v WHERE v.VIN = ? INSERT INTO
 v.Owners.SecondaryOwners VALUE ?"

    cursor = driver.execute_lambda(lambda executor:
 executor.execute_statement(statement, convert_object_to_ion(vin),

 parameter))
    logger.info('VehicleRegistration Document IDs which had secondary owners
 added: ')
    print_result(cursor)


def register_secondary_owner(driver, vin, gov_id):
    """
    Register a secondary owner for a vehicle if they are not already registered.

    :type driver: :py:class:`pyqldb.driver.qldb_driver.QldbDriver`
    :param driver: An instance of the QldbDriver class.
```

```
    :type vin: str
    :param vin: VIN of the vehicle to register a secondary owner for.

    :type gov_id: str
    :param gov_id: The government ID of the owner.
    """
    logger.info('Finding the secondary owners for vehicle with VIN:
 {}.'.format(vin))

    document_ids = get_document_id_by_gov_id(driver, gov_id)

    for document_id in document_ids:
        if is_secondary_owner_for_vehicle(driver, vin, document_id):
            logger.info('Person with ID {} has already been added as a secondary
 owner of this vehicle.'.format(gov_id))
        else:
            add_secondary_owner_for_vin(driver, vin, to_ion_struct('PersonId',
 document_id))


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Finds and adds secondary owners for a vehicle.
    """
    vin = SampleData.VEHICLE[1]['VIN']
    gov_id = SampleData.PERSON[0]['GovId']
    try:
        with create_qldb_driver(ledger_name) as driver:
            register_secondary_owner(driver, vin, gov_id)
            logger.info('Secondary owners successfully updated.')
    except Exception as e:
        logger.exception('Error adding secondary owner.')
        raise e


if __name__ == '__main__':
    main()
```

2.x

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
```

```
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
 of this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software,
 and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from pyqldbsamples.model.sample_data import to_ion_struct, get_document_ids,
 print_result, SampleData, \
    convert_object_to_ion
from pyqldbsamples.constants import Constants
from pyqldbsamples.connect_to_ledger import create_qldb_session

logger = getLogger(__name__)
basicConfig(level=INFO)


def get_document_id_by_gov_id(transaction_executor, government_id):
    """
    Find a driver's person ID using the given government ID.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type government_id: str
    :param government_id: A driver's government ID.
    :rtype: list
```

```
    :return: A list of document IDs.
    """
    logger.info("Finding secondary owner's person ID using given government ID:
 {}.".format(government_id))
    return get_document_ids(transaction_executor, Constants.PERSON_TABLE_NAME,
 'GovId', government_id)


def is_secondary_owner_for_vehicle(transaction_executor, vin,
 secondary_owner_id):
    """
    Check whether a secondary owner has already been registered for the given
 VIN.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to query.
    :type secondary_owner_id: str
    :param secondary_owner_id: The secondary owner's person ID.
    :rtype: bool
    :return: If the driver has already been registered.
    """
    logger.info('Finding secondary owners for vehicle with VIN:
 {}...'.format(vin))
    query = 'SELECT Owners.SecondaryOwners FROM VehicleRegistration AS v WHERE
 v.VIN = ?'
    rows = transaction_executor.execute_statement(query,
 convert_object_to_ion(vin))

    for row in rows:
        secondary_owners = row.get('SecondaryOwners')
        person_ids = map(lambda owner: owner.get('PersonId').text,
 secondary_owners)
        if secondary_owner_id in person_ids:
            return True
    return False


def add_secondary_owner_for_vin(transaction_executor, vin, parameter):
    """
    Add a secondary owner into `VehicleRegistration` table for a particular VIN.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
```

```
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to add a secondary owner for.
    :type parameter: :py:class:`amazon.ion.simple_types.IonPyValue`
    :param parameter: The Ion value or Python native type that is convertible to
 Ion for filling in parameters of the
                      statement.
    """
    logger.info('Inserting secondary owner for vehicle with VIN:
 {}...'.format(vin))
    statement = "FROM VehicleRegistration AS v WHERE v.VIN = '{}' INSERT INTO
 v.Owners.SecondaryOwners VALUE ?"\
        .format(vin)

    cursor = transaction_executor.execute_statement(statement, parameter)
    logger.info('VehicleRegistration Document IDs which had secondary owners
 added: ')
    print_result(cursor)


def register_secondary_owner(transaction_executor, vin, gov_id):
    """
    Register a secondary owner for a vehicle if they are not already registered.
    :type transaction_executor: :py:class:`pyqldb.execution.executor.Executor`
    :param transaction_executor: An Executor object allowing for execution of
 statements within a transaction.
    :type vin: str
    :param vin: VIN of the vehicle to register a secondary owner for.
    :type gov_id: str
    :param gov_id: The government ID of the owner.
    """
    logger.info('Finding the secondary owners for vehicle with VIN:
 {}.'.format(vin))
    document_ids = get_document_id_by_gov_id(transaction_executor, gov_id)

    for document_id in document_ids:
        if is_secondary_owner_for_vehicle(transaction_executor, vin,
 document_id):
            logger.info('Person with ID {} has already been added as a secondary
 owner of this vehicle.'.format(gov_id))
        else:
            add_secondary_owner_for_vin(transaction_executor, vin,
 to_ion_struct('PersonId', document_id))
```

```
if __name__ == '__main__':
    """
    Finds and adds secondary owners for a vehicle.
    """
    vin = SampleData.VEHICLE[1]['VIN']
    gov_id = SampleData.PERSON[0]['GovId']
    try:
        with create_qldb_session() as session:
            session.execute_lambda(lambda executor:
 register_secondary_owner(executor, vin, gov_id),
                                   lambda retry_attempt: logger.info('Retrying
 due to OCC conflict...'))
            logger.info('Secondary owners successfully updated.')
    except Exception:
        logger.exception('Error adding secondary owner.')
```

4. Para ejecutar el programa, introduzca el siguiente comando.

```
python add_secondary_owner.py
```

Para revisar estos cambios en el libro mayor `vehicle-registration`, consulte Paso 6: ver el historial de revisiones de un documento.

Paso 6: ver el historial de revisiones de un documento

Tras modificar los datos de registro de un vehículo en el paso anterior, puede consultar el historial de todos sus propietarios registrados y cualquier otro campo actualizado. En este paso, consulta el historial de revisiones de un documento de la tabla `VehicleRegistration` del libro mayor `vehicle-registration`.

Para ver el historial de revisiones

1. Utilice el siguiente programa (`query_history.py`) para consultar el historial de revisiones del documento `VehicleRegistration` con el VIN `1N4AL11D75C109151`.

   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

- Puede ver el historial de revisiones de un documento consultando la Función de historial integrada en la siguiente sintaxis.

  ```
  SELECT * FROM history( table_name [, `start-time` [, `end-time` ] ] ) AS h
  [ WHERE h.metadata.id = 'id' ]
  ```

- Tanto la hora de inicio como la hora de finalización son opcionales. Son valores literales de Amazon Ion que se pueden indicar con acentos graves (`` `...` ``). Para obtener más información, consulte Consulta de Ion con PartiQL en Amazon QLDB.

- Como práctica recomendada, califique una consulta de historial con un intervalo de fechas (hora de inicio y hora de finalización) y un identificador de documento (`metadata.id`). QLDB procesa las consultas `SELECT` en las transacciones, que están sujetas a un límite de tiempo de espera de las transacciones.

  El historial de QLDB se indexa por ID de documento y no se pueden crear índices de historial adicionales en este momento. Las consultas de historial que incluyen una hora de inicio y una hora de finalización se benefician de la calificación por intervalo de fechas.

2. Para ejecutar el programa, introduzca el siguiente comando.

```
python query_history.py
```

Para verificar criptográficamente la revisión de un documento en el libro mayor `vehicle-registration`, continúe con Paso 7: verificar un documento en un libro mayor.

Paso 7: verificar un documento en un libro mayor

Con Amazon QLDB, puede verificar de manera eficiente la integridad de un documento del diario de su libro mayor mediante el uso de hash criptográfico con SHA-256. Para obtener más información sobre cómo funcionan la verificación y el hash criptográfico en QLDB, consulte Verificación de datos en Amazon QLDB.

En este paso, verificará la revisión de un documento en la tabla `VehicleRegistration` del libro mayor `vehicle-registration`. En primer lugar, solicita un resumen, que se devuelve como un archivo de salida y actúa como firma de todo el historial de cambios del libro mayor. A continuación, solicita una prueba de la revisión relativa a ese resumen. Con esta prueba, se verifica la integridad de la revisión si se aprueban todas las comprobaciones de validación.

Para verificar la revisión de un documento

1. Revise los siguientes archivos `.py`, que representan los objetos de QLDB que se requieren para la verificación y un módulo de utilidad con funciones auxiliares para convertir los tipos de respuesta de QLDB en cadenas.

   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. Utilice dos programas `.py` (`get_digest.py` y `get_revision.py`) para realizar los siguientes pasos:

   - Solicite un nuevo resumen del libro mayor `vehicle-registration`.
   - Solicite una prueba de cada revisión del documento con el VIN `1N4AL11D75C109151` de la tabla `VehicleRegistration`.
   - Verifique las revisiones utilizando el resumen devuelto y compruébelo recalculando el resumen.

   El programa `get_digest.py` contiene el siguiente código.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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**
>
> Utilice la función `get_digest_result` para solicitar un resumen que incluya el tip actual del diario del libro mayor. La sugerencia del diario hace referencia al último bloqueo comprometido en el momento en que QLDB recibe su solicitud.

El programa `get_revision.py` contiene el siguiente código.

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**
>
> Una vez que la función `get_revision` devuelve una prueba de la revisión del documento especificado, este programa utiliza una API del lado del cliente para verificar esa revisión.

3. Para ejecutar el programa, introduzca el siguiente comando.

```
python get_revision.py
```

Si ya no necesita usar el libro mayor `vehicle-registration`, continúe con Paso 8 (opcional): limpiar recursos.

Paso 8 (opcional): limpiar recursos

Puede seguir utilizando el libro mayor `vehicle-registration`. Sin embargo, si ya no lo necesita, debe eliminarlo.

Para eliminar el libro mayor

1. Utilice el siguiente programa (`delete_ledger.py`) para eliminar el libro mayor `vehicle-registration` y todo su contenido.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
```

```
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# 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**
>
> Si la protección contra eliminación está habilitada para su libro mayor, primero debe desactivarla para poder eliminar el libro mayor utilizando la API de QLDB.

El archivo `delete_ledger.py` también depende del siguiente programa (`describe_ledger.py`).

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy,
 modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and
 to
# permit persons to whom the Software is furnished to do so.
#
```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
#
# This code expects that you have AWS credentials setup per:
# https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html
from logging import basicConfig, getLogger, INFO

from boto3 import client

from pyqldbsamples.constants import Constants

logger = getLogger(__name__)
basicConfig(level=INFO)
qldb_client = client('qldb')


def describe_ledger(ledger_name):
    """
    Describe a ledger.

    :type ledger_name: str
    :param ledger_name: Name of the ledger to describe.
    """
    logger.info('describe ledger with name: {}.'.format(ledger_name))
    result = qldb_client.describe_ledger(Name=ledger_name)
    result.pop('ResponseMetadata')
    logger.info('Success. Ledger description: {}.'.format(result))
    return result


def main(ledger_name=Constants.LEDGER_NAME):
    """
    Describe a QLDB ledger.
    """
    try:
        describe_ledger(ledger_name)
    except Exception as e:
        logger.exception('Unable to describe a ledger.')
```

```
        raise e


if __name__ == '__main__':
    main()
```

2. Para ejecutar el programa, introduzca el siguiente comando.

```
python delete_ledger.py
```

Uso de tipos de datos de Amazon Ion en Amazon QLDB

Amazon QLDB almacena sus datos en formato Amazon Ion. Para trabajar con datos en QLDB, debe usar una biblioteca de Ion como dependencia de un lenguaje de programación compatible.

En esta sección aprenderá a convertir datos de tipos nativos a sus equivalentes de Ion y viceversa. Esta guía de referencia muestra ejemplos de código que emplean el controlador QLDB para procesar datos de Ion en un libro mayor de QLDB. Incluye ejemplos de código para Java, .NET (C#), Go, Node.js (TypeScript) y Python.

Temas

Requisitos previos

Los siguientes ejemplos de código presuponen que tiene una instancia de controlador QLDB conectada a un libro mayor activo con una tabla denominada `ExampleTable`. La tabla contiene un único documento con los ocho siguientes campos:

- ExampleBool
- ExampleInt
- ExampleFloat
- ExampleDecimal
- ExampleTimestamp
- ExampleString
- ExampleBlob
- ExampleList

> **Note**
>
> A efectos de esta referencia, presuponga que el tipo almacenado en cada campo coincide con su nombre. En la práctica, QLDB no aplica definiciones de esquema o tipo de datos en los campos del documento.

Bool

En los siguientes ejemplos de código se muestra cómo procesar el tipo booleano Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

boolean exampleBoolean = driver.execute((txn) -> {
    // Transforming a Java boolean to Ion
    boolean aBoolean = true;
    IonValue ionBool = ionSystem.newBool(aBoolean);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBool = ?", ionBool);
```

```
    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBool from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java boolean
        // Cast IonValue to IonBool first
        aBoolean = ((IonBool)ionValue).booleanValue();
    }

    // exampleBoolean is now the value fetched from QLDB
    return aBoolean;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# bool to Ion.
bool nativeBool  = true;
IIonValue ionBool = valueFactory.NewBool(nativeBool);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBool = ?", ionBool);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBool from ExampleTable");
});

bool? retrievedBool = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# bool.
    retrievedBool = ionValue.BoolValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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

En los siguientes ejemplos de código se muestra cómo procesar el tipo de valor entero Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

int exampleInt = driver.execute((txn) -> {
    // Transforming a Java int to Ion
    int aInt = 256;
    IonValue ionInt = ionSystem.newInt(aInt);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleInt = ?", ionInt);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleInt from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java int
        // Cast IonValue to IonInt first
        aInt = ((IonInt)ionValue).intValue();
    }

    // exampleInt is now the value fetched from QLDB
    return aInt;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# int to Ion.
int nativeInt  = 256;
IIonValue ionInt = valueFactory.NewInt(nativeInt);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleInt = ?", ionInt);
```

```
    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleInt from ExampleTable");
});

int? retrievedInt = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# int.
    retrievedInt = ionValue.IntValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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

En los siguientes ejemplos de código se muestra cómo procesar el tipo de valor flotante Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

float exampleFloat = driver.execute((txn) -> {
    // Transforming a Java float to Ion
    float aFloat = 256;
    IonValue ionFloat = ionSystem.newFloat(aFloat);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleFloat from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java float
        // Cast IonValue to IonFloat first
        aFloat = ((IonFloat)ionValue).floatValue();
    }

    // exampleFloat is now the value fetched from QLDB
    return aFloat;
});
```

.NET

Uso de C# float

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# float to Ion.
float nativeFloat = 256;
IIonValue ionFloat = valueFactory.NewFloat(nativeFloat);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleFloat from ExampleTable");
});

float? retrievedFloat = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# float. We cast ionValue.DoubleValue to a float
    // but be cautious, this is a down-cast and can lose precision.
    retrievedFloat = (float)ionValue.DoubleValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `IResult`.

Uso de C# double

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...
```

```
IValueFactory valueFactory = new ValueFactory();

// Transforming a C# double to Ion.
double nativeDouble = 256;
IIonValue ionFloat = valueFactory.NewFloat(nativeDouble);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", ionFloat);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleFloat from ExampleTable");
});

double? retrievedDouble = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# double.
    retrievedDouble = ionValue.DoubleValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `IResult`.

Go

```
exampleFloat, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aFloat := float32(256)

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleFloat = ?", aFloat)
 if err != nil {
  return nil, err
 }
```

```
 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleFloat FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  // float64 would work as well
  var decodedResult float32
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleFloat is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonFloat(driver: QldbDriver): Promise<number> {
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Updating QLDB
            await txn.execute("UPDATE ExampleTable SET ExampleFloat = ?", 25.6);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT VALUE
 ExampleFloat FROM ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // Transforming Ion to a TypeScript Number
            const floatValue: number = ionValue.numberValue();
            return floatValue;
        })
    );
}
```

Python

```
def update_and_query_ion_float(txn):
    # QLDB can take in a Python float
    a_float = float(256)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleFloat = ?", a_float)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleFloat FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python float is a child class of Python float
        a_float = ion_value

    # example_float is now the value fetched from QLDB
    return a_float


example_float = driver.execute_lambda(lambda txn: update_and_query_ion_float(txn))
```

Decimal (Decimal)

En los siguientes ejemplos de código se muestra cómo procesar el tipo de valor decimal Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

double exampleDouble = driver.execute((txn) -> {
    // Transforming a Java double to Ion
    double aDouble = 256;
    IonValue ionDecimal = ionSystem.newDecimal(aDouble);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleDecimal = ?", ionDecimal);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleDecimal from ExampleTable");
```

```
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java double
        // Cast IonValue to IonDecimal first
        aDouble = ((IonDecimal)ionValue).doubleValue();
    }

    // exampleDouble is now the value fetched from QLDB
    return aDouble;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# decimal to Ion.
decimal nativeDecimal = 256.8723m;
IIonValue ionDecimal = valueFactory.NewDecimal(nativeDecimal);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleDecimal = ?", ionDecimal);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleDecimal from ExampleTable");
});

decimal? retrievedDecimal = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# decimal.
    retrievedDecimal = ionValue.DecimalValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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**
>
> También puede usar `ionValue.numberValue()` para transformar un decimal de Ion en un número de JavaScript. `ionValue.numberValue()` ofrece un mejor rendimiento, pero es menos preciso.

Python

```
def update_and_query_ion_decimal(txn):
    # QLDB can take in a Python decimal
    a_decimal = Decimal(256)

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleDecimal = ?", a_decimal)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleDecimal FROM ExampleTable")
```

```
    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python decimal is a child class of Python decimal
        a_decimal = ion_value

    # example_decimal is now the value fetched from QLDB
    return a_decimal


example_decimal = driver.execute_lambda(lambda txn:
 update_and_query_ion_decimal(txn))
```

Marca de tiempo

En los siguientes ejemplos de código se muestra cómo procesar el tipo de marca temporal Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

Date exampleDate = driver.execute((txn) -> {
    // Transforming a Java Date to Ion
    Date aDate = new Date();
    IonValue ionTimestamp = ionSystem.newUtcTimestamp(aDate);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleTimestamp = ?", ionTimestamp);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleTimestamp from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java Date
        // Cast IonValue to IonTimestamp first
        aDate = ((IonTimestamp)ionValue).dateValue();
    }

    // exampleDate is now the value fetched from QLDB
    return aDate;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using Amazon.IonDotnet;
...

IValueFactory valueFactory = new ValueFactory();

// Convert C# native DateTime to Ion.
DateTime nativeDateTime = DateTime.Now;

// First convert it to a timestamp object from Ion.
Timestamp timestamp = new Timestamp(nativeDateTime);
// Then convert to Ion timestamp.
IIonValue ionTimestamp = valueFactory.NewTimestamp(timestamp);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleTimestamp = ?", ionTimestamp);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleTimestamp from ExampleTable");
});

DateTime? retrievedDateTime = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# DateTime.
    retrievedDateTime = ionValue.TimestampValue.DateTimeValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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))
```

Cadena

En los siguientes ejemplos de código se muestra cómo procesar el tipo de cadena Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

String exampleString = driver.execute((txn) -> {
    // Transforming a Java String to Ion
    String aString = "Hello world!";
    IonValue ionString = ionSystem.newString(aString);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleString = ?", ionString);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleString from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java String
        // Cast IonValue to IonString first
        aString = ((IonString)ionValue).stringValue();
    }

    // exampleString is now the value fetched from QLDB
    return aString;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Convert C# string to Ion.
String nativeString = "Hello world!";
IIonValue ionString = valueFactory.NewString(nativeString);

IAsyncResult selectResult = await driver.Execute(async txn =>
```

```
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleString = ?", ionString);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleString from ExampleTable");
});

String retrievedString = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# string.
    retrievedString = ionValue.StringValue;
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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

En los siguientes ejemplos de código se muestra cómo procesar el tipo de blob Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

byte[] exampleBytes = driver.execute((txn) -> {
    // Transforming a Java byte array to Ion
    // Transform any arbitrary data to a byte array to store in QLDB
    String aString = "Hello world!";
    byte[] aByteArray = aString.getBytes();
    IonValue ionBlob = ionSystem.newBlob(aByteArray);

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleBlob from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Transforming Ion to a Java byte array
        // Cast IonValue to IonBlob first
        aByteArray = ((IonBlob)ionValue).getBytes();
    }
```

```
    // exampleBytes is now the value fetched from QLDB
    return aByteArray;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Text;
...

IValueFactory valueFactory = new ValueFactory();

// Transform any arbitrary data to a byte array to store in QLDB.
string nativeString = "Hello world!";
byte[] nativeByteArray = Encoding.UTF8.GetBytes(nativeString);
// Transforming a C# byte array to Ion.
IIonValue ionBlob = valueFactory.NewBlob(nativeByteArray);

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleBlob = ?", ionBlob);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleBlob from ExampleTable");
});

byte[] retrievedByteArray = null;

// Assume there is only one document in ExampleTable.
await foreach (IIonValue ionValue in selectResult)
{
    // Transforming Ion to a C# byte array.
    retrievedByteArray = ionValue.Bytes().ToArray();
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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))
```

Enumeración

En los siguientes ejemplos de código se muestra cómo procesar el tipo de lista Ion.

Java

```
// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

List<Integer> exampleList = driver.execute((txn) -> {
    // Transforming a Java List to Ion
    List<Integer> aList = new ArrayList<>();
    // Add 5 Integers to the List for the sake of example
    for (int i = 0; i < 5; i++) {
        aList.add(i);
    }
    // Create an empty Ion List
    IonList ionList = ionSystem.newEmptyList();
    // Add the 5 Integers to the Ion List
    for (Integer i : aList) {
        // Convert each Integer to Ion ints first to add it to the Ion List
        ionList.add(ionSystem.newInt(i));
    }

    // Insertion into QLDB
    txn.execute("UPDATE ExampleTable SET ExampleList = ?", (IonValue) ionList);

    // Fetching from QLDB
    Result result = txn.execute("SELECT VALUE ExampleList from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Iterate through the Ion List to map it to a Java List
        List<Integer> intList = new ArrayList<>();
        for (IonValue ionInt : (IonList)ionValue) {
            // Convert the 5 Ion ints to Java Integers
            intList.add(((IonInt)ionInt).intValue());
        }
        aList = intList;
    }

    // exampleList is now the value fetched from QLDB
    return aList;
});
```

.NET

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
using System.Collections.Generic;
...

IValueFactory valueFactory = new ValueFactory();

// Transforming a C# list to Ion.
IIonValue ionList = valueFactory.NewEmptyList();
foreach (int i in new List<int> {0, 1, 2, 3, 4})
{
    // Convert to Ion int and add to Ion list.
    ionList.Add(valueFactory.NewInt(i));
}

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleList = ?", ionList);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleList from ExampleTable");
});


List<int> retrievedList = new List<int>();
await foreach (IIonValue ionValue in selectResult)
{
    // Iterate through the Ion List to map it to a C# list.
    foreach (IIonValue ionInt in ionValue)
    {
        retrievedList.Add(ionInt.IntValue);
    }
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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

En QLDB, los tipos de datos `struct` son particularmente únicos respecto a otros tipos de iones. Los documentos de nivel superior que se inserten en una tabla deben ser de tipo `struct`. Un campo de documento también puede almacenar un `struct` anidado.

En aras de la simplicidad, en los siguientes ejemplos se define un documento solo con los campos `ExampleString` y `ExampleInt`.

Java

```
class ExampleStruct {
    public String exampleString;
    public int exampleInt;

    public ExampleStruct(String exampleString, int exampleInt) {
        this.exampleString = exampleString;
        this.exampleInt = exampleInt;
    }
}

// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();

ExampleStruct examplePojo = driver.execute((txn) -> {
    // Transforming a POJO to Ion
    ExampleStruct aPojo = new ExampleStruct("Hello world!", 256);
    // Create an empty Ion struct
    IonStruct ionStruct = ionSystem.newEmptyStruct();
    // Map the fields of the POJO to Ion values and put them in the Ion struct
    ionStruct.add("ExampleString", ionSystem.newString(aPojo.exampleString));
    ionStruct.add("ExampleInt", ionSystem.newInt(aPojo.exampleInt));

    // Insertion into QLDB
    txn.execute("INSERT INTO ExampleTable ?", ionStruct);

    // Fetching from QLDB
    Result result = txn.execute("SELECT * from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
```

```
    {
        // Map the fields of the Ion struct to Java values and construct a new POJO
        ionStruct = (IonStruct)ionValue;
        IonString exampleString = (IonString)ionStruct.get("ExampleString");
        IonInt exampleInt = (IonInt)ionStruct.get("ExampleInt");
        aPojo = new ExampleStruct(exampleString.stringValue(),
 exampleInt.intValue());
    }

    // examplePojo is now the document fetched from QLDB
    return aPojo;
});
```

Como alternativa, puede usar la biblioteca de Jackson para mapear tipos de datos hacia y desde Ion. La biblioteca admite los demás tipos de datos de Ion, pero este ejemplo se centra en el tipo `struct`.

```
class ExampleStruct {
    public String exampleString;
    public int exampleInt;

    @JsonCreator
    public ExampleStruct(@JsonProperty("ExampleString") String exampleString,
                         @JsonProperty("ExampleInt") int exampleInt) {
        this.exampleString = exampleString;
        this.exampleInt = exampleInt;
    }

    @JsonProperty("ExampleString")
    public String getExampleString() {
        return this.exampleString;
    }

    @JsonProperty("ExampleInt")
    public int getExampleInt() {
        return this.exampleInt;
    }
}

// Instantiate an IonSystem from the Ion library
IonSystem ionSystem = IonSystemBuilder.standard().build();
// Instantiate an IonObjectMapper from the Jackson library
IonObjectMapper ionMapper = new IonValueMapper(ionSystem);
```

```
ExampleStruct examplePojo = driver.execute((txn) -> {
    // Transforming a POJO to Ion
    ExampleStruct aPojo = new ExampleStruct("Hello world!", 256);
    IonValue ionStruct;
    try  {
        // Use the mapper to convert Java objects into Ion
        ionStruct = ionMapper.writeValueAsIonValue(aPojo);
    } catch (IOException e) {
        // Wrap the exception and throw it for the sake of simplicity in this
 example
        throw new RuntimeException(e);
    }

    // Insertion into QLDB
    txn.execute("INSERT INTO ExampleTable ?", ionStruct);

    // Fetching from QLDB
    Result result = txn.execute("SELECT * from ExampleTable");
    // Assume there is only one document in ExampleTable
    for (IonValue ionValue : result)
    {
        // Use the mapper to convert Ion to Java objects
        try {
            aPojo = ionMapper.readValue(ionValue, ExampleStruct.class);
        } catch (IOException e) {
            // Wrap the exception and throw it for the sake of simplicity in this
 example
            throw new RuntimeException(e);
        }
    }

    // examplePojo is now the document fetched from QLDB
    return aPojo;
});
```

.NET

Use la biblioteca Amazon.QLDB.Driver.Serialization para mapear los tipos de datos de C# nativos hacia y desde Ion. La biblioteca admite los demás tipos de datos de Ion, pero este ejemplo se centra en el tipo `struct`.

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

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

IAsyncQldbDriver driver = AsyncQldbDriver.Builder()
    .WithLedger("vehicle-registration")
    // Add Serialization library
    .WithSerializer(new ObjectSerializer())
    .Build();

// Creating a C# POCO.
ExampleStruct exampleStruct = new ExampleStruct
{
    ExampleString = "Hello world!",
    ExampleInt = 256
};

IAsyncResult<ExampleStruct> selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute(txn.Query<Document>("UPDATE ExampleTable SET ExampleStruct
 = ?", exampleStruct));

    // Fetching from QLDB.
    return await txn.Execute(txn.Query<ExampleStruct>("SELECT VALUE ExampleStruct
 from ExampleTable"));
});

await foreach (ExampleStruct row in selectResult)
{
    Console.WriteLine(row.ExampleString);
    Console.WriteLine(row.ExampleInt);
}
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `IResult`.

Como alternativa, puede usar la biblioteca Amazon.IonDotNet.Builders para procesar tipos de datos de Ion.

```
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;
...

IValueFactory valueFactory = new ValueFactory();

// Creating Ion struct.
IIonValue ionStruct = valueFactory.NewEmptyStruct();
ionStruct.SetField("ExampleString", valueFactory.NewString("Hello world!"));
ionStruct.SetField("ExampleInt", valueFactory.NewInt(256));

IAsyncResult selectResult = await driver.Execute(async txn =>
{
    // Insertion into QLDB.
    await txn.Execute("UPDATE ExampleTable SET ExampleStruct = ?", ionStruct);

    // Fetching from QLDB.
    return await txn.Execute("SELECT VALUE ExampleStruct from ExampleTable");
});

string retrievedString = null;
int? retrievedInt = null;

await foreach (IIonValue ionValue in selectResult)
{
    retrievedString = ionValue.GetField("ExampleString").StringValue;
    retrievedInt = ionValue.GetField("ExampleInt").IntValue;
}
```

Go

```
exampleStruct, err := driver.Execute(context.Background(), func(txn
 qldbdriver.Transaction) (interface{}, error) {
 aStruct := map[string]interface{} {
  "ExampleString": "Hello World!",
  "ExampleInt": 256,
 }

 // Insertion into QLDB
 _, err = txn.Execute("UPDATE ExampleTable SET ExampleStruct = ?", aStruct)
 if err != nil {
  return nil, err
 }
```

```
 // Fetching from QLDB
 result, err := txn.Execute("SELECT VALUE ExampleStruct FROM ExampleTable")
 if err != nil {
  return nil, err
 }

 // Assume there is only one document in ExampleTable
 if result.Next(txn) {
  var decodedResult map[string]interface{}
  err := ion.Unmarshal(result.GetCurrentData(), &decodedResult)
  if err != nil {
   return nil, err
  }

  // exampleStruct is now the value fetched from QLDB
  return decodedResult, nil
 }
 return nil, result.Err()
})
```

Node.js

```
async function queryIonStruct(driver: QldbDriver): Promise<any> {
    let exampleStruct: any = {stringValue: "Hello World!", intValue: 256};
    return (driver.executeLambda(async (txn: TransactionExecutor) => {
            // Inserting into QLDB
            await txn.execute("INSERT INTO ExampleTable ?", exampleStruct);

            // Fetching from QLDB
            const resultList: dom.Value[] = (await txn.execute("SELECT * FROM
 ExampleTable")).getResultList();

            // Assume there is only one document in ExampleTable
            const ionValue: dom.Value = resultList[0];
            // We can get all the keys of Ion struct and their associated values
            const ionFieldNames: string[] = ionValue.fieldNames();

            // Getting key and value of Ion struct to TypeScript String and Number
            const nativeStringVal: string =
 ionValue.get(ionFieldNames[0]).stringValue();
            const nativeIntVal: number =
 ionValue.get(ionFieldNames[1]).numberValue();
```

```
            // Alternatively, we can access to Ion struct fields, using their
 literal field names:
            //   const nativeStringVal = ionValue.get("stringValue").stringValue();
            //   const nativeIntVal = ionValue.get("intValue").numberValue();

            exampleStruct = {[ionFieldNames[0]]: nativeStringVal,
 [ionFieldNames[1]]: nativeIntVal};
            return exampleStruct;
        })
    );
}
```

Python

```
def update_and_query_ion_struct(txn):
    # QLDB can take in a Python struct
    a_struct = {"ExampleString": "Hello world!", "ExampleInt": 256}

    # Insertion into QLDB
    txn.execute_statement("UPDATE ExampleTable SET ExampleStruct = ?", a_struct)

    # Fetching from QLDB
    cursor = txn.execute_statement("SELECT VALUE ExampleStruct FROM ExampleTable")

    # Assume there is only one document in ExampleTable
    for ion_value in cursor:
        # Ion Python struct is a child class of Python struct
        a_struct = ion_value

    # example_struct is now the value fetched from QLDB
    return a_struct


example_struct = driver.execute_lambda(lambda txn: update_and_query_ion_struct(txn))
```

Valores nulos y tipos dinámicos

QLDB admite contenido abierto y no aplica definiciones de esquema o tipo de datos en los campos del documento. También puede almacenar valores nulos de Ion en un documento QLDB. Todos los ejemplos anteriores presuponen que cada tipo de datos devuelto es conocido y no es nulo. En los

siguientes ejemplos se muestra cómo trabajar con Ion cuando no se conoce el tipo de datos o es posible que sea nulo.

Java

```
// Empty variables
String exampleString = null;
Integer exampleInt = null;

// Assume ionValue is some queried data from QLDB
IonValue ionValue = null;

// Check the value type and assign it to the variable if it is not null
if (ionValue.getType() == IonType.STRING) {
    if (ionValue.isNullValue()) {
        exampleString = null;
    } else {
        exampleString = ((IonString)ionValue).stringValue();
    }
} else if (ionValue.getType() == IonType.INT) {
    if (ionValue.isNullValue()) {
        exampleInt = null;
    } else {
        exampleInt = ((IonInt)ionValue).intValue();
    }
};

// Creating null values
IonSystem ionSystem = IonSystemBuilder.standard().build();

// A null value still has an Ion type
IonString ionString;
if (exampleString == null) {
    // Specifically a null string
    ionString = ionSystem.newNullString();
} else {
    ionString = ionSystem.newString(exampleString);
}

IonInt ionInt;
if (exampleInt == null) {
    // Specifically a null int
    ionInt = ionSystem.newNullInt();
```

```
} else {
    ionInt = ionSystem.newInt(exampleInt);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
// The above null values still have the types 'String' and 'Int'
IonValue specialNull = ionSystem.newNull();
if (specialNull.getType() == IonType.NULL) {
    // This is true!
}
if (specialNull.isNullValue()) {
    // This is also true!
}
if (specialNull.getType() == IonType.STRING || specialNull.getType() == IonType.INT)
 {
    // This is false!
}
```

.NET

```
// Empty variables.
string exampleString = null;
int? exampleInt = null;

// Assume ionValue is some queried data from QLDB.
IIonValue ionValue;

if (ionValue.Type() == IonType.String)
{
    exampleString = ionValue.StringValue;
}
else if (ionValue.Type() == IonType.Int)
{
    if (ionValue.IsNull)
    {
        exampleInt = null;
    }
    else
    {
        exampleInt = ionValue.IntValue;
    }
};
```

```
// Creating null values.
IValueFactory valueFactory = new ValueFactory();

// A null value still has an Ion type.
IIonValue ionString = valueFactory.NewString(exampleString);

IIonValue ionInt;
if (exampleInt == null)
{
    // Specifically a null int.
    ionInt = valueFactory.NewNullInt();
}
else
{
    ionInt = valueFactory.NewInt(exampleInt.Value);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
IIonValue specialNull = valueFactory.NewNull();
if (specialNull.Type() == IonType.Null) {
    // This is true!
}
if (specialNull.IsNull) {
    // This is also true!
}
```

Go

Limitaciones de clasificación

En Go, los valores `nil` no retienen su tipo cuando los clasifica y posteriormente los desclasifica. Un valor `nil` se clasifica como Ion nulo, pero Ion nulo se desclasifica con un valor cero en lugar de `nil`.

```
ionNull, err := ion.MarshalText(nil) // ionNull is set to ion null
if err != nil {
    return
}

var result int
err = ion.Unmarshal(ionNull, &result) // result unmarshals to 0
```

```
if err != nil {
    return
}
```

Node.js

```
// Empty variables
let exampleString: string;
let exampleInt: number;

// Assume ionValue is some queried data from QLDB
// Check the value type and assign it to the variable if it is not null
if (ionValue.getType() === IonTypes.STRING) {
    if (ionValue.isNull()) {
        exampleString = null;
    } else {
        exampleString = ionValue.stringValue();
    }
} else if (ionValue.getType() === IonTypes.INT) {
    if (ionValue.isNull()) {
        exampleInt = null;
    } else {
        exampleInt = ionValue.numberValue();
    }
}

// Creating null values
if (exampleString === null) {
    ionString = dom.load('null.string');
} else {
    ionString = dom.load.of(exampleString);
}

if (exampleInt === null) {
    ionInt = dom.load('null.int');
} else {
    ionInt = dom.load.of(exampleInt);
}

// Special case regarding null!
// There is a generic null type which has Ion type 'Null'.
// The above null values still have the types 'String' and 'Int'
specialNull: dom.Value = dom.load("null.null");
```

```
if (specialNull.getType() === IonType.NULL) {
    // This is true!
}
if (specialNull.getType() === IonType.STRING || specialNull.getType() ===
 IonType.INT) {
    // This is false!
}
```

Python

```
# Empty variables
example_string = None
example_int = None

# Assume ion_value is some queried data from QLDB
# Check the value type and assign it to the variable if it is not null
if ion_value.ion_type == IonType.STRING:
    if isinstance(ion_value, IonPyNull):
        example_string = None
    else:
        example_string = ion_value
elif ion_value.ion_type == IonType.INT:
    if isinstance(ion_value, IonPyNull):
        example_int = None
    else:
        example_int = ion_value

# Creating Ion null values
if example_string is None:
    # Specifically a null string
    ion_string = loads("null.string")
else:
    # QLDB can take in Python string
    ion_string = example_string
if example_int is None:
    # Specifically a null int
    ion_int = loads("null.int")
else:
    # QLDB can take in Python int
    ion_int = example_int


# Special case regarding null!
```

```
# There is a generic null type which has Ion type 'Null'.
# The above null values still have the types 'String' and 'Int'
special_null = loads("null.null")
if special_null.ion_type == IonType.NULL:
    # This is true!
if special_null.ion_type == IonType.STRING or special_null.ion_type == IonType.INT:
    # This is false!
```

Conversión descendente a JSON

Si su aplicación requiere compatibilidad con JSON, puede realizar una conversión descendente de los datos de Amazon Ion a JSON. Sin embargo, la conversión de Ion a JSON conlleva pérdidas en algunos casos en los que los datos emplean tipos de Ion enriquecidos que no existen en JSON.

Para obtener más información sobre las reglas de conversión de Ion a JSON, consulte Conversión descendente a JSON en el Libro de recetas de Amazon Ion.

Trabajar con datos e historial en Amazon QLDB

En los siguientes temas, se proporcionan ejemplos básicos de instrucciones de creación, lectura, actualización y eliminación (CRUD). Puede ejecutar estas instrucciones manualmente mediante el editor PartiQL de la consoQLDB o el intérprete de comandos de QLDB. Esta guía también lo acompaña por el proceso de cómo QLDB gestionar sus datos a medida que realiza cambios en su libro mayor.

QLDB admite el lenguaje de consulta PartiQL.

Para ver ejemplos de código que muestran cómo ejecutar instrucciones similares mediante programación utilizando el controlador de QLDB, consulte los tutoriales en Introducción al controlador.

> **Tip**
>
> El siguiente es un breve resumen de los consejos y las mejores prácticas para trabajar con PartiQL en QLDB:
>
> - Conozca los límites de concurrencia y de transacciones: todas las instrucciones, incluidas las consultas `SELECT`, están sujetas a un control de concurrencia optimista (OCC), a límites de transacciones y conflictos, incluido un tiempo de espera de transacción de 30 segundos.
> - Utilice índices: utilice índices de cardinalidad alta y ejecute consultas dirigidas para optimizar sus instrucciones y evitar tener que escanear tablas completas. Para obtener más información, consulte Optimizar el rendimiento de las consultas.
> - Utilice predicados de igualdad: las búsquedas indexadas requieren un operador de igualdad (`=` o `IN`). Los operadores de desigualdad (`<`, `>`, `LIKE`, `BETWEEN`) no cumplen los requisitos para las búsquedas indexadas y dan como resultado escaneos de tablas completas.
> - Utilice únicamente combinaciones internas: QLDB solo admite combinaciones internas. Como práctica recomendada, combine los campos que estén indexados para cada tabla que vaya a unir. Elija índices de cardinalidad alta tanto para los criterios de unión como para los predicados de igualdad.

Temas

Crear tablas con índices e insertar documentos

Tras crear un libro mayor de Amazon QLDB, el primer paso es crear una tabla con una instrucción CREATE TABLE básica. Las tablas se componen de Documentos de QLDB, que son conjuntos de datos en formato `struct` de Amazon Ion.

Temas

Creación de tablas e índices

Las tablas tienen nombres simples que no tienen espacios de nombres. QLDB admite contenido abierto y no aplica el esquema, por lo que no se definen atributos o tipos de datos al crear tablas.

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle
```

Una instrucción `CREATE TABLE` devuelve el identificador asignado por el sistema a la nueva tabla. Todos los identificadores asignados por el sistema en QLDB son identificadores únicos universales (UUID), cada uno de los cuales se representa en una cadena codificada en Base62.

> **Note**
>
> Si lo desea, puede definir etiquetas para un recurso de tabla mientras crea la tabla. Para saber cómo hacerlo, consulte Etiquetar tablas al crearlas.

También puede crear índices en tablas para optimizar el rendimiento de las consultas.

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

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

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

> **Important**
>
> QLDB requiere un índice para buscar un documento de manera eficiente. Sin un índice, QLDB necesita escanear toda la tabla al leer los documentos. Esto puede provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
>
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad (`=` o `IN`) en un campo indexado o en un ID de documento. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Tenga en cuenta las siguientes restricciones al crear índices:

- Solo se puede crear un índice en un único campo de nivel superior. No se admiten índices compuestos, anidados, únicos ni basados en funciones.
- Puede crear un índice en cualquier tipo de datos de Ion, incluidos `list` y `struct`. Sin embargo, solo puede realizar la búsqueda indexada igualando el valor total de Ion, independientemente del

tipo de Ion. Por ejemplo, cuando se utiliza un tipo `list` como índice, no se puede realizar una búsqueda indexada por un elemento de la lista.

- El rendimiento de las consultas solo mejora cuando se utiliza un predicado de igualdad; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`.

  QLDB no respeta las desigualdades en los predicados de consulta. Como resultado, no se implementan los escaneos filtrados por rango.
- Los nombres de los campos indexados distinguen entre mayúsculas y minúsculas y pueden tener 128 caracteres como máximo.
- La creación de índices en QLDB es asíncrona. La cantidad de tiempo que tarda en crearse un índice en una tabla que no está vacía varía según el tamaño de la tabla. Para obtener más información, consulte Administrar índices.

Inserción de documentos

A continuación, puede insertar documentos en las tablas. Los documentos de QLDB se almacenan en formato Amazon Ion. Las siguientes instrucciones INSERT de PartiQL incluyen un subconjunto de los datos de muestra de registro del vehículo utilizados en Introducción a la consola de Amazon QLDB.

```
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'
} >>
```

Sintaxis y semántica de PartiQL

- Los nombres de los campos se escriben entre comillas simples (`'...'`).
- Los valores de las cadenas también se escriben entre comillas simples (`'...'`).
- Las marcas de tiempo se escriben entre acentos graves (`` `...` ``). Los acentos graves se pueden utilizar para indicar cualquier literal de Ion.
- Los enteros y los decimales son valores literales que no es necesario indicar.

Para obtener más información sobre la sintaxis y la semántica de PartiQL, consulte Consulta de Ion con PartiQL en Amazon QLDB.

Una instrucción `INSERT` crea la revisión inicial de un documento con un número de versión igual a cero. Para identificar de forma única cada documento, QLDB asigna un identificador de documento

como parte de los metadatos. Las instrucciones de inserción devuelven el identificador de cada documento que se inserta.

> **Important**
>
> Como QLDB no aplica el esquema, puede insertar el mismo documento en una tabla varias veces. Cada instrucción de inserción incluye una entrada de documento independiente en el diario y QLDB asigna a cada documento un identificador único.

Para obtener información sobre cómo consultar los documentos que ha insertado en la tabla, continúe con Consulta de sus datos.

Consulta de sus datos

La vista de usuario devuelve únicamente la última revisión no eliminada de sus datos de usuario. Esta es la vista predeterminada en Amazon QLDB. Esto significa que no se necesitan calificadores especiales si desea consultar solo sus datos.

Para obtener más información sobre la sintaxis y los parámetros de los siguientes ejemplos de consultas, consulte SELECT en la referencia de PartiQL de Amazon QLDB.

Temas

- Consultas básicas
- Proyecciones y filtros
- combinaciones;
- Datos anidados

Consultas básicas

Las consultas básicas de `SELECT` devuelven los documentos que ha insertado en la tabla.

> **Warning**
>
> Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. PartiQL admite este tipo de consultas porque es compatible con SQL. Sin embargo, no ejecute escaneados de tablas para casos de uso de producción en QLDB.

Los escaneos de tablas pueden provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad en un campo indexado o en un ID de documento, por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Las siguientes consultas muestran los resultados de los documentos de registro del vehículo que insertó previamente en Crear tablas con índices e insertar documentos. El orden de los resultados no es específico y puede variar para cada consulta `SELECT`. No debe confiar en el orden de los resultados para ninguna consulta en QLDB.

```
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**
>
> En PartiQL, se utilizan comillas simples para indicar cadenas en el lenguaje de manipulación de datos (DML) o en instrucciones de consulta. Sin embargo, la consola de QLDB y el intérprete de comandos de QLDB devuelven los resultados de las consultas en formato de texto de Amazon Ion, por lo que las cadenas aparecen entre comillas dobles.
> Esta sintaxis permite que el lenguaje de consultas de PartiQL mantenga la compatibilidad con SQL y que el formato de texto Amazon Ion mantenga la compatibilidad con JSON.

Proyecciones y filtros

Puede hacer proyecciones (dirigidas `SELECT`) y otros filtros estándar (cláusulas `WHERE`). La siguiente consulta devuelve un subconjunto de campos de documentos de la tabla `VehicleRegistration`. Filtra los vehículos según los siguientes criterios:

- Filtro de cadenas: está registrado en Seattle.

- Filtro decimal: tiene un importe de penalización pendiente inferior a `100.0`.
- Filtro de fecha: tiene una fecha de registro válida a partir del 4 de septiembre de 2019 incluido.

```
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" }]
    }
}
```

combinaciones;

También puede escribir consultas de combinación internas. El siguiente ejemplo muestra una consulta de combinación interna implícita que devuelve todos los documentos de registro junto con los atributos de los vehículos registrados.

```
SELECT * FROM VehicleRegistration AS r, Vehicle AS v
WHERE r.VIN = v.VIN
AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFromDate: 2017-08-21T,
    ValidToDate: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
```

```
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    },
    Type: "Sedan",
    Year: 2011,
    Make: "Audi",
    Model: "A5",
    Color: "Silver"
},
{
    VIN: "KM8SRDHF6EU074761",
    LicensePlateNumber: "CA762X",
    State: "WA",
    City: "Kent",
    PendingPenaltyTicketAmount: 130.75,
    ValidFromDate: 2017-09-14T,
    ValidToDate: 2020-06-25T,
    Owners: {
        PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
        SecondaryOwners: []
    },
    Type: "Sedan",
    Year: 2015,
    Make: "Tesla",
    Model: "Model S",
    Color: "Blue"
}
```

O bien, puede escribir la misma consulta de combinación interna con la siguiente sintaxis explícita.

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

Datos anidados

Puede usar PartiQL en QLDB para consultar datos anidados en documentos. En el siguiente ejemplo, se muestra una subconsulta correlacionada que aplana datos anidados. En este caso, el carácter `@` es técnicamente opcional. Sin embargo, indica de forma explícita que desea la estructura `Owners` dentro de `VehicleRegistration` y no una recopilación diferente denominada `Owners` (si existiera).

```
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: []
}
```

A continuación, se muestra una subconsulta de la lista SELECT que proyecta datos anidados, además de una combinación interna.

```
SELECT
    v.Make,
    v.Model,
    (SELECT VALUE o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM
    VehicleRegistration AS r, Vehicle AS v
WHERE
    r.VIN = v.VIN AND r.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

```
{
    Make: "Audi",
    Model: "A5",
    PrimaryOwner: ["294jJ3YUoH1IEEm8GSabOs"]
},
{
    Make: "Tesla",
    Model: "Model S",
    PrimaryOwner: ["IN7MvYtUjkp1GMZu0F6CG9"]
}
```

La siguiente consulta devuelve `PersonId` y el número de índice (ordinal) de cada persona de la lista `Owners.SecondaryOwners` de un documento `VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = '1N4AL11D75C109151'
```

```
{
    PersonId: "5Ufgdlnj06gF5CWcOIu64s",
    owner_idx: 0
}
```

Para obtener información sobre cómo consultar los metadatos del documento, continúe con Consulta de los metadatos del documento.

Consulta de los metadatos del documento

Una instrucción `INSERT` crea la revisión inicial de un documento con un número de versión igual a cero. Para identificar de forma única cada documento, Amazon QLDB asigna un identificador de documento como parte de los metadatos.

Además del identificador del documento y el número de versión, QLDB almacena otros metadatos generados por el sistema para cada documento de una tabla. Estos metadatos incluyen información sobre las transacciones, los atributos del diario y el valor hash del documento.

Todos los identificadores asignados por el sistema en QLDB son identificadores únicos universales (UUID), cada uno de los cuales se representa en una cadena codificada en Base62. Para obtener más información, consulte Identificadores únicos en Amazon QLDB.

Temas

- Vista confirmada
- Combinar las vistas confirmadas y de usuario

Vista confirmada

Puede acceder a los metadatos del documento consultando la vista confirmada. Esta vista devuelve documentos de la tabla generada por el sistema que corresponde directamente a su tabla de usuario. Incluye la última revisión confirmada y no eliminada de sus datos y de los metadatos generados por el sistema. Para consultar esta vista, añada el prefijo `_ql_committed_` al nombre de la tabla en la consulta. (El prefijo `_ql_` está reservado en QLDB para los objetos del sistema).

```
SELECT * FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

Utilizando los datos previamente insertados en Crear tablas con índices e insertar documentos, el resultado de esta consulta muestra el contenido del sistema de la última revisión de cada documento no eliminado. El documento del sistema tiene los metadatos anidados en el campo `metadata` y los datos de usuario anidados en el campo `data`.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{wCsmM6qD4STxz0WYmE+47nZvWtcCz9D6zNtCiM5GoWg=}},
    data:{
        VIN: "1N4AL11D75C109151",
        LicensePlateNumber: "LEWISR261LL",
        State: "WA",
        City: "Seattle",
        PendingPenaltyTicketAmount: 90.25,
        ValidFromDate: 2017-08-21T,
        ValidToDate: 2020-05-11T,
        Owners: {
            PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
            SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
        }
    },
    metadata:{
        id:"3Qv67yjXEwB9SjmvkuG6Cp",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
},
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:14
    },
    hash:{{wPuwH60TtcCvg/23BFp+redRXuCALkbDihkEvCX22Jk=}},
    data:{
        VIN: "KM8SRDHF6EU074761",
```

```
        LicensePlateNumber: "CA762X",
        State: "WA",
        City: "Kent",
        PendingPenaltyTicketAmount: 130.75,
        ValidFromDate: 2017-09-14T,
        ValidToDate: 2020-06-25T,
        Owners: {
            PrimaryOwner: { PersonId: "IN7MvYtUjkp1GMZu0F6CG9" },
            SecondaryOwners: []
        }
    },
    metadata:{
        id:"JOzfB3lWqGU727mpPeWyxg",
        version:0,
        txTime:2019-06-05T20:53:321d-3Z,
        txId:"HgXAkLjAtV0HQ4lNYdzX60"
    }
}
```

Campos de la vista confirmada

- `blockAddress`: la ubicación del bloque del diario de su libro mayor en el que se confirmó la revisión del documento. Una dirección, que se puede utilizar para la verificación criptográfica, tiene los dos campos siguientes.
  - `strandId`: el identificador único de la cadena del diario que contiene el bloque.
  - `sequenceNo`: un número de índice que especifica la ubicación del bloque dentro de la cadena.

  > **Note**
  >
  > Los dos documentos de este ejemplo tienen un `blockAddress` idéntico con el mismo `sequenceNo`. Como estos documentos se insertaron en una sola transacción (y en este caso, en una sola instrucción), se confirmaron en el mismo bloque.

- `hash`: el valor hash de Ion SHA-256 que representa de forma única la revisión del documento. El hash cubre los campos `data` y `metadata` de la revisión y se puede utilizar para la verificación criptográfica.
- `data`: los atributos de datos de usuario del documento.

  Si redacta una revisión, esta estructura `data` se sustituye por un campo `dataHash` cuyo valor es el hash de Ion de la estructura `data` eliminada.

- `metadata`: los atributos de metadatos del documento.
  - `id`: el identificador único asignado por el sistema al documento.
  - `version`: el número de versión del documento. Se trata de un entero de base cero que se incrementa con cada revisión del documento.
  - `txTime`: la fecha y hora de confirmación de la revisión del documento en el diario.
  - `txId`: el ID único de la transacción que confirmó la revisión del documento.

Combinar las vistas confirmadas y de usuario

Puede escribir consultas que combinen una tabla de la vista confirmada con una tabla de la vista de usuario. Por ejemplo, puede que desee combinar el documento `id` de una tabla con un campo definido por el usuario de otra tabla.

La siguiente consulta combina dos tablas denominadas `DriversLicense` y `Person` en sus campos `PersonId` e `id` de documento respectivamente, utilizando la vista confirmada para esta última.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS p
ON d.PersonId = p.metadata.id
WHERE p.metadata.id = '1CWScY2qHYI9G88C2SjvtH'
```

Para obtener información sobre cómo consultar el campo de identificador del documento en la vista de usuario predeterminada, continúe con Uso de la cláusula BY para consultar el identificador del documento.

Uso de la cláusula BY para consultar el identificador del documento

Si bien puede definir campos concebidos para ser identificadores únicos (por ejemplo, el número de chasis [VIN] de un vehículo), el verdadero identificador único de un documento es el campo de metadatos `id`, como se describe en Inserción de documentos. Por este motivo, puede utilizar el campo `id` para crear relaciones entre tablas.

Solo se puede acceder directamente al campo `id` del documento en la vista confirmada, pero también puede proyectarlo en la vista de usuario predeterminada mediante la cláusula BY. Para ver un ejemplo, repase la siguiente consulta y sus resultados.

```
SELECT r_id, r.VIN, r.LicensePlateNumber, r.State, r.City, r.Owners
```

```
FROM VehicleRegistration AS r BY r_id
WHERE r_id = '3Qv67yjXEwB9SjmvkuG6Cp'
```

```
{
    r_id: "3Qv67yjXEwB9SjmvkuG6Cp",
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

En esta consulta, `r_id` es un alias definido por el usuario que se establece en la cláusula `FROM` mediante la palabra clave `BY`. Este alias `r_id` se une con el campo de metadatos `id` de cada documento del conjunto de resultados de la consulta. Puede usar este alias en la cláusula `SELECT` y también en la cláusula `WHERE` de una consulta en la vista de usuario.

Sin embargo, para acceder a otros atributos de metadatos, debe consultar la vista confirmada.

Combinar con el identificador de documento

Suponga que utiliza el documento `id` de una tabla como clave externa en un campo definido por el usuario de otra tabla. Puede usar la cláusula `BY` para escribir una consulta de combinación interna para las dos tablas de estos campos (similar a Combinar las vistas confirmadas y de usuario del tema anterior).

El siguiente ejemplo combina dos tablas denominadas `DriversLicense` y `Person` en sus campos `PersonId` e `id` de documento respectivamente, utilizando la cláusula `BY` para esta última.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = '1CWScY2qHYI9G88C2SjvtH'
```

Para obtener información sobre cómo realizar cambios en un documento de la tabla, continúe con Actualizar y eliminar documentos.

Actualizar y eliminar documentos

En Amazon QLDB, la revisión de un documento es una estructura Amazon Ion que representa una versión única de una secuencia de documentos identificados mediante un ID de documento único. Cada revisión contiene el conjunto de datos completo del documento, incluidos los datos de usuario y los metadatos generados por el sistema. Cada revisión se identifica de forma única mediante una combinación del identificador del documento y un número de versión de base cero.

Al actualizar un documento, QLDB crea una nueva revisión con el mismo identificador de documento y un número de versión incrementado. El ciclo de vida de un documento finaliza cuando se elimina de una tabla. Esto significa que no se puede volver a crear ninguna revisión de documento con el mismo identificador de documento.

Realizar revisiones de documentos

Por ejemplo, las siguientes instrucciones insertan un registro de vehículo nuevo, actualizan la ciudad de registro y, a continuación, eliminan el registro. Esto da como resultado tres revisiones de un documento.

```
INSERT INTO VehicleRegistration
{
    'VIN' : '1HVBBAANXWH544237',
    'LicensePlateNumber' : 'LS477D',
    'State' : 'WA',
    'City' : 'Tacoma',
    'PendingPenaltyTicketAmount' : 42.20,
    'ValidFromDate' : `2011-10-26T`,
    'ValidToDate' : `2023-09-25T`,
    'Owners' : {
        'PrimaryOwner' : { 'PersonId': 'KmA3XPKKFqYCP2zhR3d0Ho' },
        'SecondaryOwners' : []
    }
}
```

> **Note**
>
> Las instrucciones insertar y otras instrucciones de DML devuelven el identificador de cada documento afectado. Antes de continuar, guarde este identificador porque lo necesita para la función de historial del tema siguiente. También puede obtener el ID de documento con la siguiente consulta.

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

```
UPDATE VehicleRegistration AS r
SET r.City = 'Bellevue'
WHERE r.VIN = '1HVBBAANXWH544237'
```

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Para obtener más ejemplos e información sobre la sintaxis de estas instrucciones de DML, consulte UPDATE y DELETE en la referencia de PartiQL de Amazon QLDB.

Para insertar y eliminar elementos específicos de un documento, puede utilizar instrucciones `UPDATE` u otras instrucciones de DML que comiencen con la palabra clave `FROM`. Consulte la referencia FROM (INSERT, REMOVE o SET) para obtener más información y ejemplos.

Tras eliminar un documento, ya no podrá consultarlo en las vistas confirmadas o de usuario. Para obtener información sobre cómo consultar el historial de revisiones de este documento mediante la función de historial integrada, continúe con Consultar el historial de revisiones.

Consultar el historial de revisiones

Amazon QLDB almacena el historial completo de todos los documentos de una tabla. Para ver las tres revisiones del documento de registro del vehículo que insertó, actualizó y eliminó anteriormente en Actualizar y eliminar documentos, consulte la función de historial integrada.

Temas

- Función de historial
- Ejemplo de consulta de historial

Función de historial

La función de historial de QLDB es una extensión PartiQL que devuelve las revisiones de la vista de la tabla definida por el sistema. Por lo tanto, incluye sus datos y los metadatos asociados en el mismo esquema que la vista confirmada.

Sintaxis

```
SELECT * FROM history( table_name | 'table_id' [, `start-time` [, `end-time` ] ] ) AS h
[ WHERE h.metadata.id = 'id' ]
```

Argumentos

table_name | '***table_id***'

El nombre de la tabla o el identificador de la tabla. El nombre de una tabla es un identificador PartiQL que puede indicarse con comillas dobles o sin comillas. Un identificador de tabla es un literal de cadena que se debe incluir entre comillas simples. Para obtener más información sobre cómo usar identificadores de tabla, consulte Consultar el historial de tablas inactivas.

`` `start-time` ``, `` `end-time` ``

(Opcional) Especifica el intervalo de tiempo durante el que estuvieron activas las revisiones. Estos parámetros no especifican el intervalo de tiempo durante el cual las revisiones se confirmaron en el diario de una transacción.

Las horas de inicio y finalización son literales de marca temporal de Ion que se pueden indicar con acentos graves (`` `...` ``). Para obtener más información, consulte Consulta de Ion con PartiQL en Amazon QLDB.

Estos parámetros de tiempo funcionan de la siguiente manera:

- Tanto la hora de inicio como la hora de finalización están incluidas. Deben estar en formato de fecha y hora ISO 8601 y en hora universal coordinada (UTC).
- La hora de inicio debe ser anterior o igual a la hora de finalización y puede ser cualquier fecha pasada arbitraria.
- La hora de finalización debe ser inferior o igual a la fecha y hora UTC actuales.
- Si especifica una hora de inicio, pero no una hora de finalización, la consulta establece de forma predeterminada la hora de finalización en la fecha y hora actuales. Si no especifica ninguna de las dos, la consulta devolverá el historial completo.

'*id*'

(Opcional) El identificador del documento cuyo historial de revisiones desea consultar, indicado entre comillas simples.

> **Tip**
>
> Como práctica recomendada, califique una consulta de historial con un intervalo de fechas (hora de inicio y hora de finalización) y un identificador de documento (`metadata.id`). En QLDB, cada consulta `SELECT` se procesa en una transacción y está sujeta a un límite de tiempo de espera de la transacción.
> Las consultas de historial no utilizan los índices que crea en una tabla. El historial de QLDB solo se indexa por ID de documento y no se pueden crear índices de historial adicionales de momento. Las consultas de historial que incluyen una hora de inicio y una hora de finalización se benefician de la calificación por intervalo de fechas.

Ejemplo de consulta de historial

Para consultar el historial del documento de registro de vehículos, utilice el `id` que guardó anteriormente en Actualizar y eliminar documentos. Por ejemplo, la siguiente consulta de historial devuelve cualquier revisión del identificador del documento `ADR2Ll1fGsU4Jr4EqTdnQF` que haya estado activa entre `2019-06-05T00:00:00Z` y `2019-06-05T23:59:59Z`.

> **Note**
>
> Los parámetros de hora de inicio y finalización no especifican el intervalo de tiempo durante el cual las revisiones se confirmaron en el diario de una transacción. Por ejemplo, si una revisión se confirmó antes de `2019-06-05T00:00:00Z` y permaneció activa después de esa hora de inicio, esta consulta de ejemplo mostrará esa revisión en los resultados.

Asegúrese de sustituir `id`, la hora de inicio y la hora de finalización por sus propios valores, según proceda.

```
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
```

Los resultados deben ser similares a los siguientes.

```
{
    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"
    }
}
```

El resultado incluye atributos de metadatos que proporcionan detalles sobre cuándo se modificó cada elemento y mediante qué transacción. A partir de estos datos, puede comprobar lo siguiente:

- El documento se identifica de forma única por su `id` asignado por el sistema: `ADR2Ll1fGsU4Jr4EqTdnQF`. Se trata de un identificador único universal (UUID) que se representa en una cadena codificada en Base62.
- Una instrucción `INSERT` crea la revisión inicial de un documento (versión `0`).
- Cada actualización posterior crea una nueva revisión con el mismo documento `id` y un número de versión incrementado.
- El campo `txId` indica la transacción que confirmó cada revisión y `txTime` muestra cuándo se confirmó cada una.
- Una instrucción `DELETE` crea una revisión nueva, pero final, de un documento. Esta revisión final solo tiene metadatos.

Para obtener información sobre cómo eliminar una revisión de forma permanente, continúe con Editar revisiones de documentos.

Editar revisiones de documentos

En Amazon QLDB, una instrucción `DELETE` solo elimina un documento de forma lógica al crear una nueva revisión que lo marca como eliminado. QLDB también permite realizar una operación de edición de datos para eliminar permanentemente las revisiones de documentos inactivos del historial de una tabla.

> **Note**
>
> Cualquier libro mayor que se haya creado antes del 22 de julio de 2021 actualmente no es apto para la edición. Puede ver la hora de creación de su libro mayor en la consola de Amazon QLDB.

La operación de edición elimina solo los datos de usuario de la revisión especificada, sin alterar la secuencia ni los metadatos del documento. Esto mantiene la integridad general de los datos del libro mayor.

Antes de empezar con la redacción de datos en QLDB, asegúrese de revisar Condiciones y limitaciones de edición en la referencia de PartiQL de Amazon QLDB.

Temas

- Procedimiento almacenado de edición
- Comprobar si una edición está completa
- Ejemplo de edición
- Eliminar y editar una revisión activa
- Editar un campo concreto de una revisión

Procedimiento almacenado de edición

Puede utilizar el procedimiento almacenado REDACT_REVISION para eliminar permanentemente una revisión individual e inactiva de un libro mayor. Este procedimiento almacenado elimina todos los datos de usuario de la revisión especificada tanto en el almacenamiento indexado como en el del

diario. Sin embargo, no modifica la secuencia del diario y los metadatos del documento, incluidos el identificador del documento y el hash. La operación es irreversible.

La revisión del documento especificada debe ser una revisión inactiva en el historial. La última revisión activa de un documento no es apta para ser editada.

Para editar varias revisiones, debe ejecutar el procedimiento almacenado una vez para cada revisión. Puede editar una revisión por transacción.

Sintaxis

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argumentos

`*block-address*`

La ubicación en el bloque del diario de la revisión del documento que se va a editar. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Se trata de un valor literal de Ion que se indica con acentos graves. Por ejemplo:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

'*table-id*'

El identificador único de la tabla cuya revisión del documento desea editar, indicado entre comillas simples.

'*document-id*'

El identificador único del documento cuya revisión desea editar, indicado entre comillas simples.

Comprobar si una edición está completa

Cuando envía una solicitud de edición ejecutando el procedimiento almacenado, QLDB procesa la edición de los datos de forma asíncrona. Al finalizar, los datos de usuario de la revisión (representados por la estructura `data`) se eliminan permanentemente. Para comprobar si una solicitud de edición se ha completado, puede utilizar una de las siguientes opciones:

- Exportación de diario
- Secuencia de diario
- Operación de la API GetBlock
- Operación de la API GetRevision
- Función de historial – Nota: Una vez completada una edición en el diario, puede pasar algún tiempo antes de que las consultas del historial muestren el resultado de la edición. Es posible que vea algunas revisiones editadas antes que otras a medida que se completa la edición asíncrona, pero las consultas del historial acabarán mostrando los resultados finalizados.

Una vez finalizada la edición de una revisión, la estructura `data` de la revisión se sustituye por un campo `dataHash` nuevo. El valor de este campo es el hash de Ion de la estructura `data` eliminada, como muestra el siguiente ejemplo. Como resultado, el libro mayor mantiene la integridad general de sus datos y sigue siendo verificable criptográficamente mediante las operaciones de la API de verificación existentes. Para obtener más información sobre la verificación, consulte Verificación de datos en Amazon QLDB.

Ejemplo de edición

Tome el documento de registro del vehículo que revisó anteriormente en Consultar el historial de revisiones. Suponga que desea editar la segunda revisión (`version:1`). El siguiente ejemplo de consulta muestra esta revisión antes de la edición. En los resultados de la consulta, la estructura `data` que se editará aparece resaltada en *cursiva roja*.

```
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"
    }
}
```

Tome nota de `blockAddress` en los resultados de la consulta porque debe pasar este valor al procedimiento `REDACT_REVISION` almacenado. A continuación, busque el identificador único de la tabla `VehicleRegistration` consultando el catálogo del sistema, de la siguiente manera.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Use este identificador de la tabla junto con el identificador del documento y la dirección de bloqueo para ejecutar `REDACT_REVISION`. El identificador de la tabla y el identificador del documento son literales de cadena que deben escribirse entre comillas simples, y la dirección de bloque es un literal de Ion entre acentos graves. Asegúrese de sustituir estos argumentos por sus propios valores, según proceda.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

> **Tip**
>
> Cuando utiliza la consola de QLDB o el intérprete de comandos de QLDB para consultar un identificador de tabla o un identificador de documento (o cualquier valor literal de cadena), el valor devuelto aparece entre comillas dobles. Sin embargo, al especificar los argumentos del identificador de la tabla y el identificador del documento del procedimiento `REDACT_REVISION` almacenado, debe escribir los valores entre comillas simples.

Esto se debe a que usted escribe instrucciones en formato PartiQL, pero QLDB devuelve los resultados en formato Amazon Ion. Para obtener más información sobre la sintaxis y la semántica de PartiQL en QLDB, consulte Consulta de Ion con PartiQL.

Una solicitud de edición válida devuelve una estructura de Ion que representa la revisión del documento que está editando, de la siguiente manera.

```
{
  blockAddress: {
    strandId: "JdxjkR9bSYB5jMHWcI464T",
    sequenceNo: 17
  },
  tableId: "5PLf9SXwndd63lPaSIa0O6",
  documentId: "ADR2Ll1fGsU4Jr4EqTdnQF",
  version: 1
}
```

Cuando ejecuta el procedimiento almacenado, QLDB procesa la solicitud de edición de forma asíncrona. Al finalizar la edición, la estructura `data` se elimina permanentemente y se sustituye por un campo nuevo *`dataHash`*. El valor de este campo es el hash de Ion de la estructura `data` eliminada.

> **Note**
>
> Este ejemplo de `dataHash` se proporciona únicamente con fines informativos y no es un valor hash real calculado.

```
{
    blockAddress:{
        strandId:"JdxjkR9bSYB5jMHWcI464T",
        sequenceNo:17
    },
    hash:{{LGSFZ4iEYWZeMwmAqcxxNyT4wbCtuMOmFCj8pEd6Mp0=}},
    dataHash: {{s83jd7sfhsdfhksj7hskjdfjfpIPP/DP2hvionas2d4=}},
    metadata:{
        id:"ADR2Ll1fGsU4Jr4EqTdnQF",
        version:1,
        txTime:2019-06-05T21:01:442d-3Z,
```

```
        txId:"9cArhIQV5xf5Tf5vtsPwPq"
    }
}
```

Eliminar y editar una revisión activa

Las revisiones de documentos activos (es decir, las últimas revisiones no eliminadas de cada documento) no son aptas para la edición de datos. Para poder editar una revisión activa, primero debe actualizarla o eliminarla. Esto mueve la revisión previamente activa al historial y la habilita para su edición.

Si su caso de uso requiere que todo el documento se marque como eliminado, utilice primero una instrucción DELETE. Por ejemplo, la siguiente instrucción elimina de forma lógica el documento `VehicleRegistration` con un VIN de `1HVBBAANXWH544237`.

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

A continuación, edite la revisión anterior antes de eliminarla, tal y como se describió anteriormente. Si es necesario, también puede editar individualmente cualquier revisión anterior.

Si su caso de uso requiere que el documento permanezca activo, utilice primero una instrucción UPDATE o FROM para ocultar o eliminar los campos que desee editar. Este proceso se describe en las secciones siguientes.

Editar un campo concreto de una revisión

QLDB no admite la edición de un campo en particular dentro de la revisión de un documento. Para ello, primero puede utilizar una instrucción UPDATE-REMOVE o FROM-REMOVE para eliminar un campo existente de una revisión. Por ejemplo, la siguiente instrucción elimina el campo `LicensePlateNumber` del documento `VehicleRegistration` con un VIN de `1HVBBAANXWH544237`.

```
UPDATE VehicleRegistration AS r
REMOVE r.LicensePlateNumber
WHERE r.VIN = '1HVBBAANXWH544237'
```

A continuación, edite la revisión anterior antes de eliminarla, tal y como se describió anteriormente. Si es necesario, también puede editar individualmente cualquier revisión anterior que incluya este campo ahora eliminado.

Para obtener información sobre cómo optimizar sus consultas, continúe con Optimizar el rendimiento de las consultas.

Optimizar el rendimiento de las consultas

El objetivo de Amazon QLDB es abordar las necesidades de las cargas de trabajo de procesamiento de transacciones online (OLTP) de alto rendimiento. Esto significa que QLDB está optimizada para un conjunto específico de patrones de consulta, aunque admite capacidades de consulta similares a las de SQL. Es fundamental diseñar las aplicaciones y sus modelos de datos para que funcionen con estos patrones de consulta. De lo contrario, a medida que las tablas crezcan, se producirán importantes problemas de rendimiento, como la latencia de las consultas, los tiempos de espera de las transacciones y los conflictos de concurrencia.

Esta sección describe las restricciones de consulta en QLDB y guía sobre cómo escribir consultas óptimas dadas estas restricciones.

Temas

Límite de tiempo de espera de transacción

En QLDB, cada instrucción de PartiQL (incluida cada consulta `SELECT`) se procesa en una transacción y está sujeta a un límite de tiempo de espera de la transacción. Una transacción puede ejecutarse durante un máximo de 30 segundos antes de confirmarse. Tras este límite, QLDB rechaza cualquier trabajo realizado en la transacción y descarta la sesión que ejecutó la transacción. Este límite evita que el cliente de sesión pierda sesiones al iniciar transacciones y no confirmarlas ni cancelarlas.

Conflictos de concurrencia

QLDB implementa el control de concurrencia mediante el control de concurrencia optimista (OCC). Las consultas subóptimas también pueden provocar más conflictos de OCC. Para obtener más información acerca de OCC, consulte Modelo de concurrencia de Amazon QLDB.

Patrones de consulta óptimos

Como práctica recomendada, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` que filtre por campo indexado o por identificador de documento. QLDB requiere un operador de igualdad (= o `IN`) en un campo indexado para buscar un documento de manera eficiente.

Los siguientes son ejemplos de patrones de consulta óptimos en la vista de usuario.

```
--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'
```

Cualquier consulta que no siga estos patrones invoca un escaneado completo de la tabla. Los escaneos de tablas pueden provocar tiempos de espera de transacción para consultas en tablas grandes o consultas que devuelven conjuntos de resultados de gran tamaño. También pueden provocar conflictos de OCC con transacciones competidoras.

Índices de alta cardinalidad

Se recomienda indexar los campos que contienen valores de cardinalidad alta. Por ejemplo, los campos `VIN` y `LicensePlateNumber` de la tabla `VehicleRegistration` son campos indexados concebidos para ser únicos.

Evite indexar campos de baja cardinalidad, como códigos de estado, direcciones, estados o provincias y códigos postales. Si indexa un campo de este tipo, las consultas pueden producir conjuntos de resultados grandes que tienen más probabilidades de provocar tiempos de espera en las transacciones o provocar conflictos de OCC no deseados.

Consultas de vista confirmada

Las consultas que se ejecutan en la vista confirmada siguen las mismas pautas de optimización que las consultas de vista de usuario. Los índices que se crean en una tabla también se utilizan para las consultas en la vista confirmada.

Consultas de la función historial

Las consultas de la función historial no utilizan los índices creados en una tabla. El historial de QLDB solo se indexa por ID de documento y no se pueden crear índices de historial adicionales de momento.

Como práctica recomendada, califique una consulta de historial con un intervalo de fechas (hora de inicio y hora de finalización) y un identificador de documento (`metadata.id`). Las consultas de historial que incluyen una hora de inicio y una hora de finalización se benefician de la calificación por intervalo de fechas.

Consultas de Ion internas

Para las consultas de combinación internas, utilice criterios de combinación que incluyan al menos un campo indexado para la tabla del lado derecho de la combinación. Sin un índice de combinación, una consulta de combinación invoca varios escaneos de tabla: para cada documento de la tabla izquierda de la combinación, la consulta escanea completamente la tabla derecha. La práctica recomendada consiste en combinar los campos que estén indexados para cada tabla a la que vaya a combinar, además de especificar un predicado de igualdad `WHERE` para al menos una tabla.

Por ejemplo, la siguiente consulta combina las tablas `VehicleRegistration` y `Vehicle` de sus campos `VIN` respectivos, ambos indexados. Esta consulta también tiene un predicado de igualdad en `VehicleRegistration.VIN`.

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

Seleccione índices de cardinalidad alta tanto para los criterios de combinación como para los predicados de igualdad en sus consultas de combinación.

Patrones de consulta que deben evitarse

Los siguientes son algunos ejemplos de instrucciones subóptimas que no se escalan bien para tablas más grandes en QLDB. Le recomendamos encarecidamente que no confíe en este tipo de consultas para las tablas que crecen con el tiempo, ya que sus consultas acabarán provocando

tiempos de espera de las transacciones. Como las tablas contienen documentos que varían en tamaño, es difícil definir límites precisos para las consultas no indexadas.

```
--No predicate clause
SELECT * FROM Vehicle

--COUNT() is not an optimized function
SELECT COUNT(*) FROM Vehicle

--Low-cardinality predicate
SELECT * FROM Vehicle WHERE Color = 'Silver'

--Inequality (>) does not qualify for indexed lookup
SELECT * FROM Vehicle WHERE "Year" > 2019

--Inequality (LIKE)
SELECT * FROM Vehicle WHERE VIN LIKE '1N4AL%'

--Inequality (BETWEEN)
SELECT SUM(PendingPenaltyTicketAmount) FROM VehicleRegistration
WHERE ValidToDate BETWEEN `2020-01-01T` AND `2020-07-01T`

--No predicate clause
DELETE FROM Vehicle

--No document id, and no date range for the history() function
SELECT * FROM history(Vehicle)
```

En general, no recomendamos ejecutar los siguientes tipos de patrones de consulta para los casos de uso de producción en QLDB:

- Consultas de procesamiento analítico en línea (OLAP)
- Consultas exploratorias sin cláusula de predicado
- Consultas de informes
- Text search

En su lugar, recomendamos transmitir los datos a un servicio de base de datos personalizada y optimizado para casos de uso analíticos. Por ejemplo, puede transmitir datos de QLDB a Amazon OpenSearch Service para ofrecer capacidades de búsqueda de texto completo en los documentos. Para ver un ejemplo de aplicación que demuestre este caso de uso, consulte el repositorio de GitHub

aws-samples/amazon-qldb-streaming-amazon-opensearch-service-sample-python. Para obtener información acerca de secuencias QLDB, consulte Transmisión de datos de diarios desde Amazon QLDB.

Monitorear el desempeño

El controlador de QLDB proporciona información sobre el uso de E/S consumido y la temporización en el objeto resultado de una instrucción. Puede utilizar estas métricas para identificar instrucciones PartiQL ineficientes. Para obtener más información, consulte Obtener estadísticas de instrucciones PartiQL.

También puede utilizar Amazon CloudWatch para realizar un seguimiento del rendimiento de su libro mayor para las operaciones de datos. Supervise la métrica `CommandLatency` para un `LedgerName` y `CommandType` específicos. Para obtener más información, consulte Monitorización con Amazon CloudWatch. Para obtener información sobre cómo QLDB utiliza los comandos para administrar las operaciones de datos, consulte Gestión de sesiones con el controlador.

Obtener estadísticas de instrucciones PartiQL

Amazon QLDB proporciona estadísticas de ejecución de instrucciones que lo ayudarán a optimizar el uso de Amazon QLDB mediante la ejecución de instrucciones PartiQL más eficientes. QLDB devuelve estas estadísticas junto con los resultados de la instrucción. Incluyen métricas que cuantifican el uso de E/S consumido y el tiempo de procesamiento del lado del servidor, que puede utilizar para identificar instrucciones ineficientes.

Esta característica está disponible actualmente en el editor de PartiQL de la consola de QLDB, en el intérprete de comandos de QLDB y en la última versión del controlador de QLDB para todos los lenguajes compatibles. También puede ver las estadísticas de las instrucciones para su historial de consultas en la consola.

Temas

- Uso de E/S
- Información de temporización

Uso de E/S

La métrica de uso de E/S describe la cantidad de solicitudes de E/S de lectura. Si el número de solicitudes de E/S de lectura es superior al esperado, indica que la instrucción no está optimizada,

por ejemplo, porque no hay un índice. Le recomendamos que consulte Patrones de consulta óptimos en el tema anterior, Optimización del rendimiento de las consultas.

> **Note**
>
> Al ejecutar una instrucción `CREATE INDEX` en una tabla que no esté vacía, la métrica de uso de E/S incluye únicamente las solicitudes de lectura de la llamada de creación de índices sincrónica.
> QLDB crea el índice de todos los documentos existentes en la tabla de forma asíncrona. Estas solicitudes de lectura asíncronas no se incluyen en la métrica de uso de E/S de los resultados de su instrucción. Las solicitudes de lectura asincrónica se cargan por separado y se añaden al total de E/S de lectura una vez finalizada la creación del índice.

Usar la consola de QLDB

Para obtener el uso de E/S de lectura de una instrucción mediante la consola de QLDB, lleve a cabo los siguientes pasos:

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, seleccione Editor PartiQL.
3. Seleccione un libro mayor de la lista desplegable de libros mayores.
4. En la ventana del editor de consultas, introduzca cualquier instrucción que desee y, a continuación, seleccione Ejecutar. A continuación se muestra un ejemplo de consulta:

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Para ejecutar una instrucción, también puede utilizar el atajo de teclado Ctrl+Enter para Windows o Cmd+Return para macOS. Para obtener más atajos de teclado, consulte Atajos de teclado del editor PartiQL.
5. Debajo de la ventana del editor de consultas, los resultados de la consulta incluyen las E/S de lectura, que es el número de solicitudes de lectura realizadas por la instrucción.

También puede ver las E/S de lectura del historial de consultas siguiendo estos pasos:

1. En el panel de navegación, seleccione Consultas recientes en el editor de PartiQL.

2. En la columna E/S de lectura, se muestra el número de solicitudes de lectura realizadas por cada instrucción.

Uso del controlador de QLDB

Para obtener el uso de E/S de una instrucción mediante el controlador de QLDB, llame a la operación `getConsumedIOs` del cursor de secuencia o del cursor almacenado en búfer del resultado.

En los siguientes ejemplos de código, se muestra cómo obtener E/S de lectura del cursor de secuencia del resultado de una instrucción.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.IOUsage;
import software.amazon.qldb.Result;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

driver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);

    for (IonValue ionValue : result) {
        // User code here to handle results
    }

    IOUsage ioUsage = result.getConsumedIOs();
    long readIOs = ioUsage.getReadIOs();
});
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

// This is one way of creating Ion values. We can also use a ValueFactory.
```

```
// For more details, see: https://docs.aws.amazon.com/qldb/latest/developerguide/
driver-cookbook-dotnet.html#cookbook-dotnet.ion
IIonValue ionFirstName = IonLoader.Default.Load("Jim");

await driver.Execute(async txn =>
{
    IAsyncResult result = await txn.Execute("SELECT * FROM testTable WHERE firstName
 = ?", ionFirstName);

    // Iterate through stream cursor to accumulate read IOs.
    await foreach (IIonValue ionValue in result)
    {
        // User code here to handle results.
        // Warning: It is bad practice to rely on results within a lambda block,
 unless
        // it is to check the state of a result. This is because lambdas are
 retryable.
    }

    var ioUsage = result.GetConsumedIOs();
    var readIOs = ioUsage?.ReadIOs;
});
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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))
```

En los siguientes ejemplos de código, se muestra cómo obtener E/S de lectura del cursor almacenado en búfer del resultado de una instrucción. Esto devuelve el total de E/S de lectura de las solicitudes `ExecuteStatement` y `FetchPage`.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.IOUsage;
import software.amazon.qldb.Result;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

Result result = driver.execute(txn -> {
    return txn.execute("SELECT * FROM testTable WHERE firstName = ?", ionFirstName);
});

IOUsage ioUsage = result.getConsumedIOs();
long readIOs = ioUsage.getReadIOs();
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);
});

var ioUsage = result.GetConsumedIOs();
var readIOs = ioUsage?.ReadIOs;
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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**
>
> El cursor de secuencia tiene estado porque pagina el conjunto de resultados. Por lo tanto, las operaciones `getConsumedIOs` y `getTimingInformation` devuelven las métricas acumuladas desde el momento en que las llama.
> El cursor almacenado en búfer almacena el conjunto de resultados en la memoria y devuelve el total de métricas acumuladas.

Información de temporización

La métrica de información de temporización describe el tiempo de procesamiento del lado del servidor en milisegundos. El tiempo de procesamiento del lado del servidor se define como la cantidad de tiempo que QLDB dedica a procesar una instrucción. Esto no incluye el tiempo dedicado a las llamadas o pausas de la red. Esta métrica elimina la ambigüedad entre el tiempo de procesamiento en el lado del servidor de QLDB y el tiempo de procesamiento en el lado del cliente.

Usar la consola de QLDB

Para obtener información de temporización de una instrucción mediante la consola de QLDB, lleve a cabo los siguientes pasos:

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, seleccione Editor PartiQL.
3. Seleccione un libro mayor de la lista desplegable de libros mayores.

4. En la ventana del editor de consultas, introduzca cualquier instrucción que desee y, a continuación, seleccione Ejecutar. A continuación se muestra un ejemplo de consulta:

   ```
   SELECT * FROM testTable WHERE firstName = 'Jim'
   ```

   Para ejecutar una instrucción, también puede utilizar el atajo de teclado Ctrl+Enter para Windows o Cmd+Return para macOS. Para obtener más atajos de teclado, consulte Atajos de teclado del editor PartiQL.
5. Debajo de la ventana del editor de consultas, los resultados de la consulta incluyen la latencia del lado del servidor, que es el tiempo transcurrido entre el momento en que QLDB recibe la solicitud de instrucción y el momento en que envía la respuesta. Es un subconjunto de la duración total de la consulta.

También puede ver la información de temporización de su historial de consultas siguiendo estos pasos:

1. En el panel de navegación, seleccione Consultas recientes en el editor de PartiQL.
2. La columna Tiempo de ejecución (ms) muestra esta información de temporización para cada instrucción.

Uso del controlador de QLDB

Para obtener información de temporización de una instrucción mediante el controlador de QLDB, llame a la operación `getTimingInformation` del curos de secuencia o almacenado en búfer del resultado.

En los siguientes ejemplos de código, se muestra cómo obtener el tiempo de procesamiento del cursor de secuencia del resultado de una instrucción.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.Result;
import software.amazon.qldb.TimingInformation;

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

```
IonValue ionFirstName = ionSystem.newString("Jim");

driver.execute(txn -> {
    Result result = txn.execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);

    for (IonValue ionValue : result) {
        // User code here to handle results
    }

    TimingInformation timingInformation = result.getTimingInformation();
    long processingTimeMilliseconds =
 timingInformation.getProcessingTimeMilliseconds();
});
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

await driver.Execute(async txn =>
{
    IAsyncResult result = await txn.Execute("SELECT * FROM testTable WHERE firstName
 = ?", ionFirstName);

    // Iterate through stream cursor to accumulate processing time.
    await foreach(IIonValue ionValue in result)
    {
        // User code here to handle results.
        // Warning: It is bad practice to rely on results within a lambda block,
 unless
        // it is to check the state of a result. This is because lambdas are
 retryable.
    }

    var timingInformation = result.GetTimingInformation();
    var processingTimeMilliseconds = timingInformation?.ProcessingTimeMilliseconds;
});
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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))
```

En los siguientes ejemplos de código, se muestra cómo obtener el tiempo de procesamiento del cursor almacenado en búfer del resultado de una instrucción. Esto devuelve el tiempo de procesamiento total de las solicitudes `ExecuteStatement` y `FetchPage`.

Java

```
import com.amazon.ion.IonSystem;
import com.amazon.ion.IonValue;
import com.amazon.ion.system.IonSystemBuilder;
import software.amazon.qldb.Result;
import software.amazon.qldb.TimingInformation;

IonSystem ionSystem = IonSystemBuilder.standard().build();
IonValue ionFirstName = ionSystem.newString("Jim");

Result result = driver.execute(txn -> {
    return txn.execute("SELECT * FROM testTable WHERE firstName = ?", ionFirstName);
});
```

```
TimingInformation timingInformation = result.getTimingInformation();
long processingTimeMilliseconds = timingInformation.getProcessingTimeMilliseconds();
```

.NET

```
using Amazon.IonDotnet.Builders;
using Amazon.IonDotnet.Tree;
using Amazon.QLDB.Driver;
using IAsyncResult = Amazon.QLDB.Driver.IAsyncResult;

IIonValue ionFirstName = IonLoader.Default.Load("Jim");

IAsyncResult result = await driver.Execute(async txn =>
{
    return await txn.Execute("SELECT * FROM testTable WHERE firstName = ?",
 ionFirstName);
});

var timingInformation = result.GetTimingInformation();
var processingTimeMilliseconds = timingInformation?.ProcessingTimeMilliseconds;
```

> **Note**
>
> Para convertirlo a código sincrónico, elimine las palabras clave `await` y `async` y cambie el tipo `IAsyncResult` a `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**
>
> El cursor de secuencia tiene estado porque pagina el conjunto de resultados. Por lo tanto, las operaciones `getConsumedIOs` y `getTimingInformation` devuelven las métricas acumuladas desde el momento en que las llama.

El cursor almacenado en búfer almacena el conjunto de resultados en la memoria y devuelve el total de métricas acumuladas.

Para obtener más información sobre cómo consultar el catálogo del sistema, continúe con Consulta del catálogo del sistema.

Consulta del catálogo del sistema

Cada tabla que cree en un libro mayor de Amazon QLDB tiene un identificador único asignado por el sistema. Puede encontrar el identificador de una tabla, su lista de índices y otros metadatos consultando la tabla del catálogo del sistema `information_schema.user_tables`.

Todos los identificadores asignados por el sistema en QLDB son identificadores únicos universales (UUID), cada uno de los cuales se representa en una cadena codificada en Base62. Para obtener más información, consulte Identificadores únicos en Amazon QLDB.

En el siguiente ejemplo, se muestran los resultados de una consulta que devuelve los atributos de metadatos de la tabla `VehicleRegistration`.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

```
{
    tableId: "5PLf9SXwndd63lPaSIa0O6",
    name: "VehicleRegistration",
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

Campos de metadatos de la tabla

- `tableId`: el ID exclusivo de la tabla.
- `name`: el nombre de la tabla.
- `indexes`: La lista de índices de la tabla.

  - `indexId`: el ID exclusivo del índice.
  - `expr`: la ruta del documento indexado. Este campo es una cadena en formato: `[fieldName]`.
  - `status`: el estado actual del índice (`BUILDING`, `FINALIZING`, `ONLINE`, `FAILED` o `DELETING`). QLDB no utiliza el índice en las consultas hasta que el estado es `ONLINE`.
  - `message`: el mensaje de error que describe el motivo por el que el índice tiene un estado `FAILED`. Este campo solo se incluye para índices fallidos.
- `status`: el estado actual de la tabla (`ACTIVE` o `INACTIVE`). Una tabla se convierte en `INACTIVE` cuando ejecuta `DROP`.

Para aprender a gestionar tablas mediante las instrucciones `DROP TABLE` y `UNDROP TABLE`, continúe con Administrar tablas.

Administrar tablas

En esta sección, se describe cómo administrar tablas utilizando las instrucciones `DROP TABLE` y `UNDROP TABLE` de Amazon QLDB. También se describe cómo etiquetar las tablas mientras las crea. Las cuotas para el número de tablas activas y el total de tablas que puede crear se definen en Cuotas y límites de Amazon QLDB.

Temas

- Etiquetar tablas al crearlas
- Eliminar tablas
- Consultar el historial de tablas inactivas
- Reactivar tablas

Etiquetar tablas al crearlas

> **Note**
>
> El etiquetado de tablas al crearlas solo se admite en los libros mayores en el modo de permisos `STANDARD`.

Puede etiquetar los recursos de tabla. Para administrar las etiquetas de las tablas existentes, utilice la AWS Management Console o las operaciones `TagResource`, `UntagResource` y

`ListTagsForResource` de la API. Para obtener más información, consulte Etiquetado de recursos de Amazon QLDB.

También puede definir etiquetas de tabla mientras crea la tabla mediante la consola de QLDB o especificándolas en una instrucción `CREATE TABLE` de PartiQL. En el siguiente ejemplo, se crea una tabla llamada `Vehicle` con la etiqueta `environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Al etiquetar los recursos en el momento de su creación, ya no es necesario ejecutar scripts de etiquetado personalizados después de la creación del recurso. Una vez etiquetada una tabla, puede controlar el acceso a la tabla según esas etiquetas. Por ejemplo, puede conceder acceso total solo a las tablas que tengan una etiqueta específica. Para ver una política de ejemplo JSON, consulte Acceso completo a todas las acciones basadas en las etiquetas de las tablas.

Eliminar tablas

Para eliminar una tabla, use una instrucción DROP TABLE básica. Cuando elimina una tabla en QLDB, simplemente la está desactivando.

Por ejemplo, la siguiente instrucción desactiva la tabla `VehicleRegistration`.

```
DROP TABLE VehicleRegistration
```

Una instrucción `DROP TABLE` devuelve el identificador asignado por el sistema a la tabla. El estado `VehicleRegistration` ahora debería ser `INACTIVE` en la tabla del catálogo del sistema information_schema.user_tables.

```
SELECT status FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

Consultar el historial de tablas inactivas

Además del nombre de una tabla, también puede consultar Función de historial de QLDB con un identificador de tabla como primer argumento de entrada. Debe usar el identificador de la tabla para consultar el historial de una tabla inactiva. Una vez desactivada una tabla, ya no podrá consultar su historial con el nombre de la tabla.

En primer lugar, busque el ID de la tabla consultando la tabla del catálogo del sistema. Por ejemplo, la siguiente consulta devuelve el `tableId` de la tabla `VehicleRegistration`.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

A continuación, puede usar este ID para ejecutar la misma consulta de historial desde Consultar el historial de revisiones. A continuación, se muestra un ejemplo en el que se consulta el historial del identificador del documento `ADR2Ll1fGsU4Jr4EqTdnQF` a partir del identificador de la tabla `5PLf9SXwndd63lPaSIa0O6`. El identificador de tabla es un literal de cadena que se debe incluir entre comillas simples.

```
--replace both the table and document IDs with your values
SELECT * FROM history('5PLf9SXwndd63lPaSIa0O6', `2000T`, `2019-06-05T23:59:59Z`) AS h
WHERE h.metadata.id = 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Reactivar tablas

Luego de desactivar una tabla en QLDB, puede utilizar la instrucción UNDROP TABLE para reactivarla.

En primer lugar, busque el identificador de la tabla en `information_schema.user_tables`. Por ejemplo, la siguiente consulta devuelve el `tableId` de la tabla `VehicleRegistration`. El estado debería ser `INACTIVE`.

```
SELECT tableId FROM information_schema.user_tables
WHERE name = 'VehicleRegistration'
```

A continuación, utilice este identificador para reactivar la tabla. A continuación, se muestra un ejemplo que anula la acción de descartar el ID de tabla `5PLf9SXwndd63lPaSIa0O6`. En este caso, el identificador de la tabla es un identificador único que se escribe entre comillas dobles.

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

El estado de `VehicleRegistration` ahora debería ser `ACTIVE`.

Para obtener información sobre cómo crear, describir y eliminar índices, continúe con Administrar índices.

Administrar índices

En esta sección, se describe cómo crear, describir y eliminar índices en Amazon QLDB. La cuota del número de índices por tabla que puede crear se define en Cuotas y límites de Amazon QLDB.

Temas

Crear índices

Como también se describe en Creación de tablas e índices, puede usar la instrucción CREATE INDEX para crear un índice en una tabla para un campo de nivel superior específico, de la siguiente manera. Tanto el nombre de la tabla como el nombre del campo indexado distinguen entre mayúsculas y minúsculas.

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

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

Cada índice que cree en una tabla tiene un identificador único asignado por el sistema. Para encontrar este identificador de índice, consulte la siguiente sección Describir índices.

> **Important**
>
> QLDB requiere un índice para buscar un documento de manera eficiente. Sin un índice, QLDB necesita escanear toda la tabla al leer los documentos. Esto puede provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
>
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad (`=` o `IN`) en un campo indexado o en un ID de documento. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Tenga en cuenta las siguientes restricciones al crear índices:

- Solo se puede crear un índice en un único campo de nivel superior. No se admiten índices compuestos, anidados, únicos ni basados en funciones.
- Puede crear un índice en cualquier tipo de datos de Ion, incluidos `list` y `struct`. Sin embargo, solo puede realizar la búsqueda indexada igualando el valor total de Ion, independientemente del tipo de Ion. Por ejemplo, cuando se utiliza un tipo `list` como índice, no se puede realizar una búsqueda indexada por un elemento de la lista.
- El rendimiento de las consultas solo mejora cuando se utiliza un predicado de igualdad; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`.

  QLDB no respeta las desigualdades en los predicados de consulta. Como resultado, no se implementan los escaneos filtrados por rango.
- Los nombres de los campos indexados distinguen entre mayúsculas y minúsculas y pueden tener 128 caracteres como máximo.
- La creación de índices en QLDB es asíncrona. La cantidad de tiempo que tarda en crearse un índice en una tabla que no está vacía varía según el tamaño de la tabla. Para obtener más información, consulte Administrar índices.

Describir índices

La creación de índices en QLDB es asíncrona. La cantidad de tiempo que tarda en crearse un índice en una tabla que no está vacía varía según el tamaño de la tabla. Para comprobar el estado de la creación de un índice, puede consultar la tabla del catálogo del sistema information_schema.user_tables.

Por ejemplo, la siguiente instrucción consulta todos los índices de la tabla `VehicleRegistration` en el catálogo del sistema.

```
SELECT VALUE indexes
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration'
```

```
{
    indexId: "Djg2nt0yIs2GY0T29Kud1z",
    expr: "[VIN]",
    status: "ONLINE"
},
```

```
{
    indexId: "4tPW3fUhaVhDinRgKRLhGU",
    expr: "[LicensePlateNumber]",
    status: "FAILED",
    message: "aws.ledger.errors.InvalidEntityError: Document contains multiple values
 for indexed field: LicensePlateNumber"
}
```

Campos de índice

- `indexId`: el ID exclusivo del índice.
- `expr`: la ruta del documento indexado. Este campo es una cadena en formato: `[fieldName]`.
- `status`: el estado actual del índice. El estado puede ser uno de los siguientes valores:
  - `BUILDING`: está creando activamente el índice de la tabla.
  - `FINALIZING`: ha terminado de crear el índice y está empezando a activarlo para su uso.
  - `ONLINE`: está activo y listo para usarse en consultas. QLDB no utiliza el índice en las consultas hasta que el estado es en línea.
  - `FAILED`: no puede crear el índice debido a un error irrecuperable. Los índices en este estado siguen contando para su cuota de índices por tabla. Para obtener más información, consulte Errores comunes.
  - `DELETING`: elimina activamente el índice después de que un usuario lo haya descartado.
- `message`: el mensaje de error que describe el motivo por el que el índice tiene un estado `FAILED`. Este campo solo se incluye para índices fallidos.

Con la consola

También puede utilizar el AWS Management Console para comprobar el estado de un índice.

Para comprobar el estado de un índice (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, seleccione Libros mayores.
3. En la lista de libros mayores, seleccione el nombre del libro mayor cuyos índices desee administrar.
4. En la página de detalles del libro mayor, en la pestaña Tablas, seleccione el nombre de la tabla cuyo índice desee comprobar.

5. En la página de detalles de la tabla, busque la tarjeta Campos indexados. La columna Estado del índice muestra el estado actual de cada índice de la tabla.

Eliminar índices

Use la instrucción DROP INDEX para descartar un índice. Cuando descarta un índice, se elimina permanentemente de la tabla.

En primer lugar, busque el identificador del índice en `information_schema.user_tables`. Por ejemplo, la siguiente consulta devuelve `indexId` del campo `LicensePlateNumber` indexado de la tabla `VehicleRegistration`.

```
SELECT indexes.indexId
FROM information_schema.user_tables info, info.indexes indexes
WHERE info.name = 'VehicleRegistration' and indexes.expr = '[LicensePlateNumber]'
```

A continuación, utilice este identificador para descartar el índice. A continuación, se muestra un ejemplo en el que se descarta el identificador del índice `4tPW3fUhaVhDinRgKRLhGU`. En este caso, el identificador del índice es un identificador único que se escribe entre comillas dobles.

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

> **Note**
>
> La cláusula `WITH (purge = true)` es obligatoria para todas las instrucciones `DROP INDEX` y actualmente `true` es el único valor admitido.
> La palabra clave `purge` distingue entre mayúsculas y minúsculas y debe escribirse completamente en minúsculas.

Con la consola

También puede utilizar AWS Management Console para descartar un índice.

Para descartar un índice (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.

2. En el panel de navegación, seleccione Libros mayores.
3. En la lista de libros mayores, seleccione el nombre del libro mayor cuyos índices desee administrar.
4. En la página de detalles del libro mayor, en la pestaña Tablas, elija el nombre de la tabla cuyo índice desee descartar.
5. En la página de detalles de la tabla, busque la tarjeta Campos indexados. Seleccione el índice que desee descartar y, a continuación, elija Descartar índice.

Errores comunes

En esta sección, se describen los errores más frecuentes que se pueden encontrar al crear índices y se sugieren posibles soluciones.

> **Note**
>
> Los índices con estado `FAILED` siguen contando para su cuota de índices por tabla. Un índice erróneo también impide modificar o eliminar cualquier documento de la tabla que haya provocado un error en la creación del índice.
> Debe descartar el índice de forma explícita para eliminarlo de la cuota.

El documento contiene varios valores para el campo indexado: ***`fieldName`***.

QLDB no puede crear un índice para el nombre de campo especificado porque la tabla contiene un documento con varios valores para el mismo campo (es decir, nombres de campo duplicados).

Primero debe descartar el índice fallido. A continuación, asegúrese de que todos los documentos de la tabla tengan solo un valor para cada nombre de campo antes de volver a intentar crear el índice. También puede crear un índice para otro campo que no tenga duplicados.

QLDB también devuelve este error si intenta insertar un documento que contenga varios valores para un campo que ya está indexado en la tabla.

Se ha superado el límite de índices: la tabla ***`tableName`*** ya tiene ***`n`*** índices y no puede crear más.

QLDB impone un límite de cinco índices por tabla, incluidos los índices fallidos. Debe descartar un índice existente antes de crear uno nuevo.

No hay un índice definido con el identificador: ***indexID***.

Intentó eliminar un índice que no existe para la combinación especificada de tabla e identificador de índice. Para obtener información sobre cómo comprobar los índices existentes, consulte Describir índices.

Identificadores únicos en Amazon QLDB

En esta sección, se describen las propiedades y las pautas de uso de los identificadores únicos asignados por el sistema en Amazon QLDB. También proporciona algunos ejemplos de identificadores únicos de QLDB.

Temas

- Propiedades
- Uso
- Ejemplos

Propiedades

Todos los identificadores asignados por el sistema en QLDB son identificadores únicos universales (UUID). Cada identificador incluye las siguientes propiedades:

- Número de UUID de 128 bits
- Representado en texto codificado en Base62
- Cadena alfanumérica de 22 caracteres de longitud fija (por ejemplo: `3Qv67yjXEwB9SjmvkuG6Cp`)

Uso

Cuando utilice identificadores únicos de QLDB en su aplicación, tenga en cuenta las siguientes pautas:

Debe

- Tratar el identificador como una cadena.

No debe

- Intentar decodificar la cadena.
- Atribuir un significado semántico a la cadena (por ejemplo, derivar un componente temporal).
- Ordenar las cadenas en orden semántico.

Ejemplos

Los siguientes atributos son algunos ejemplos de identificadores únicos de QLDB:

- ID del documento
- ID de índice
- ID de cadena
- ID de la tabla
- ID de transacción

Modelo de concurrencia de Amazon QLDB

El objetivo de Amazon QLDB es abordar las necesidades de las cargas de trabajo de procesamiento de transacciones online (OLTP) de alto rendimiento. QLDB admite capacidades de consulta similares a las de SQL y ofrece transacciones ACID completas. Además, los elementos de datos de QLDB son documentos que ofrecen flexibilidad de esquema y un modelado de datos intuitivo. Con un diario como base, puede usar QLDB para acceder al historial completo y verificable de todos los cambios en sus datos y transmitir transacciones coherentes a otros servicios de datos según sea necesario.

Temas

Control de concurrencia

QLDB implementa el control de concurrencia mediante el control de concurrencia optimista (OCC). El OCC se basa en el principio de que varias transacciones pueden completarse con frecuencia sin interferir entre sí.

Al usar OCC, las transacciones en la QLDB no adquieren bloqueos en los recursos de la base de datos y funcionan con un aislamiento serializable total. QLDB ejecuta transacciones simultáneas en serie, de modo que produce el mismo efecto que si esas transacciones se hubieran iniciado en serie.

Antes de confirmarse, cada transacción realiza una comprobación de validación para garantizar que ninguna otra transacción confirmada haya modificado los datos a los que está accediendo. Si esta comprobación revela modificaciones contradictorias o si el estado de los datos cambia, se rechaza la transacción de confirmación. Sin embargo, la transacción se puede reiniciar.

Cuando una transacción se escribe en QLDB, las comprobaciones de validación del modelo OCC las implementa la propia QLDB. Si no se puede escribir una transacción en el diario debido a un error en la fase de verificación de la OCC, QLDB devuelve una `OccConflictException` a la capa de

aplicación. El software de la aplicación es responsable de garantizar que la transacción se reinicie. La aplicación debería abortar la transacción rechazada y, a continuación, volver a intentar toda la transacción desde el principio.

Para obtener información sobre cómo el controlador QLDB gestiona y reintenta los conflictos de OCC y otras excepciones transitorias, consulte Descripción de la política de reintentos del controlador en Amazon QLDB.

Uso de índices para evitar escanear tablas completas

En la QLDB, cada instrucción PartiQL (incluidas las consultas `SELECT`) se procesa en una transacción y está sujeta a un límite de tiempo de espera de la transacción.

Como práctica recomendada, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` que filtre un campo indexado o un identificador de documento. QLDB requiere un operador de igualdad en un campo indexado para buscar un documento de manera eficiente; por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`.

Sin esta búsqueda indexada, la QLDB necesita escanear toda la tabla al leer los documentos. Esto puede provocar una latencia en las consultas y tiempos de espera de las transacciones, además de aumentar las probabilidades de que la OCC entre en conflicto con transacciones competidoras.

Por ejemplo, piense en una tabla llamada `Vehicle` que solo tiene un índice en el campo `VIN`. Contiene los siguientes documentos.

VIN	Make	Modelo	Color
`"1N4AL11D 75C109151"`	`"Audi"`	`"A5"`	`"Silver"`
`"KM8SRDHF 6EU074761"`	`"Tesla"`	`"Model S"`	`"Blue"`
`"3HGGK5G5 3FM761765"`	`"Ducati"`	`"Monster 1200"`	`"Yellow"`
`"1HVBBAAN XWH544237"`	`"Ford"`	`"F 150"`	`"Black"`

VIN	Make	Modelo	Color
"1C4RJFAG 0FC625797"	"Mercedes"	"CLK 350"	"White"

Dos usuarios simultáneos llamados Alice y Bob trabajan con la misma tabla en un libro mayor. Quieren actualizar dos documentos diferentes, de la siguiente manera.

Alice:

```
UPDATE Vehicle AS v
SET v.Color = 'Blue'
WHERE v.VIN = '1N4AL11D75C109151'
```

Bob:

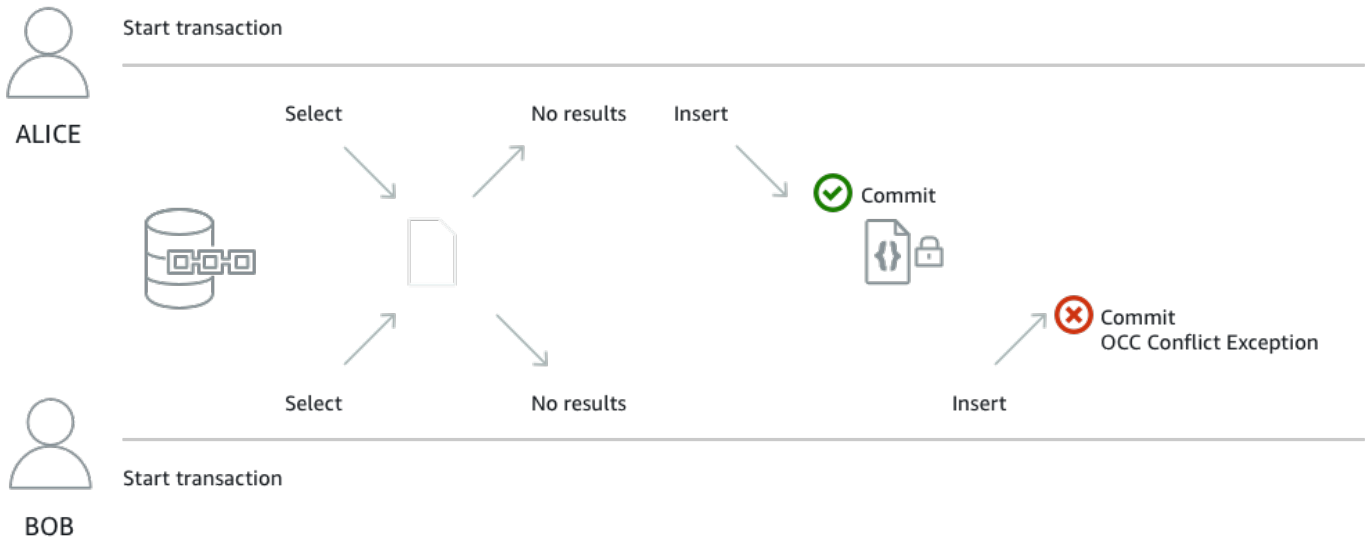
```
UPDATE Vehicle AS v
SET v.Color = 'Red'
WHERE v.Make = 'Tesla' AND v.Model = 'Model S'
```

Supongamos que Alice y Bob inician sus transacciones al mismo tiempo. La instrucción `UPDATE` de Alice realiza una búsqueda indexada en el campo `VIN`, por lo que solo necesita leer ese documento. Alice termina y confirma correctamente su transacción primero.

La instrucción de Bob filtra los campos no indexados, por lo que escanea una tabla y encuentra un `OccConflictException`. Esto se debe a que la transacción confirmada por Alice modificó los datos a los que accede la instrucción de Bob, que incluyen todos los documentos de la tabla, no solo los documentos que Bob está actualizando.

Conflictos de OCC de inserción

Los conflictos de OCC pueden incluir documentos recién insertados, no solo documentos que ya existían anteriormente. Fíjese en el siguiente diagrama, en el que dos usuarios (Alice y Bob) trabajan con el mismo elemento de una tabla en un libro mayor. Ambos quieren insertar un documento nuevo solo con la condición de que aún no exista un valor predicado.



En este ejemplo, tanto Alice como Bob ejecutan las siguientes instrucciones `SELECT` y `INSERT` en una sola transacción. Su aplicación ejecuta la instrucción `INSERT` solo si la instrucción `SELECT` no devuelve ningún resultado.

```
SELECT * FROM Vehicle v WHERE v.VIN = 'ABCDE12345EXAMPLE'
```

```
INSERT INTO Vehicle VALUE
{
    'VIN' : 'ABCDE12345EXAMPLE',
    'Type' : 'Wagon',
    'Year' : 2019,
    'Make' : 'Subaru',
    'Model' : 'Outback',
    'Color' : 'Gray'
}
```

Supongamos que Alice y Bob inician sus transacciones al mismo tiempo. Sus dos consultas `SELECT` no devuelven ningún documento existente con un `VIN` de `ABCDE12345EXAMPLE`. Por lo tanto, sus solicitudes continúan con la instrucción `INSERT`.

Alice termina y confirma correctamente su transacción primero. Entonces, Bob intenta confirmar su transacción, pero QLDB la rechaza y lanza una `OccConflictException`. Esto se debe a que la transacción confirmada por Alice modificó el conjunto de resultados de la consulta `SELECT` de Bob y el OCC detecta este conflicto antes de confirmar la transacción de Bob.

La consulta `SELECT` es necesaria para que este ejemplo de transacción sea idempotente. A continuación, Bob puede volver a intentar toda la transacción desde el principio. Pero su siguiente consulta `SELECT` devolverá el documento que Alice insertó, por lo que la solicitud de Bob no ejecutará `INSERT`.

Hacer que las transacciones sean idempotentes

La transacción de inserción de la sección anterior también es un ejemplo de transacción idempotente. En otras palabras, ejecutar la misma transacción varias veces produce resultados idénticos. Si Bob ejecuta `INSERT` sin comprobar primero si un determinado `VIN` ya existe, es posible que la tabla acabe con documentos con valores `VIN` duplicados.

Considere otros escenarios de reintentos además de los conflictos de OCC. Supongamos que QLDB confirma correctamente la transacción en el lado del servidor pero el cliente agota el tiempo de espera mientras espera una respuesta. Le recomendamos que haga que las transacciones de escritura sean idempotentes para evitar cualquier efecto secundario inesperado en caso de reintentos.

Conflictos de OCC de edición

QLDB evita la edición simultánea de revisiones en el mismo bloque de diario. Piense en un ejemplo en el que dos usuarios simultáneos (Alice y Bob) quieren editar dos revisiones de documentos diferentes que estén consignadas en el mismo bloque de un libro mayor. En primer lugar, Alice solicita la edición de una revisión ejecutando el procedimiento almacenado `REDACT_REVISION`, tal como se indica a continuación.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '5PLf9SXwndd63lPaSIa0O6', 'ADR2Ll1fGsU4Jr4EqTdnQF'
```

Luego, mientras la solicitud de Alice aún se está procesando, Bob solicita la edición de otra revisión, de la siguiente forma.

```
EXEC REDACT_REVISION `{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

QLDB rechaza la solicitud de Bob con una `OccConflictException` a pesar de que están intentando editar dos revisiones de documentos diferentes. Esto se debe a que la revisión de Bob se

encuentra en el mismo bloque que la revisión que Alice está editando. Cuando la solicitud de Alice termine de procesarse, Bob podrá volver a intentar su solicitud de edición.

Del mismo modo, si dos transacciones simultáneas intentan editar la misma revisión, solo se podrá procesar una solicitud. La otra solicitud falla con una excepción de conflicto de OCC hasta que se complete la edición. Posteriormente, cualquier solicitud de edición de la misma revisión generará un error que indicará que la revisión ya está editada.

Administración de las sesiones de concurrencia

Si tiene experiencia en el uso de un sistema de administración de base de datos relacional (RDBMS), puede que esté familiarizado con las conexiones simultáneas. QLDB no tiene el mismo concepto de conexión RDBMS tradicional, ya que las transacciones se ejecutan con mensajes de solicitud y respuesta HTTP.

El concepto análogo en QLDB es el de sesión activa. Conceptualmente, una sesión es similar al inicio de sesión de un usuario: gestiona la información sobre sus solicitudes de transacción de datos a un libro mayor. Una sesión activa es aquella en la que se ejecuta una transacción de forma activa. También puede tratarse de una sesión en la que se ha finalizado recientemente una transacción, y el servicio prevé iniciar otra transacción de forma inmediata. QLDB admite una transacción en ejecución activa por sesión.

El límite de sesiones activas simultáneas por libro mayor se define en Cuotas y límites de Amazon QLDB. Una vez alcanzado este límite, cualquier sesión que intente iniciar una transacción dará como resultado un error (`LimitExceededException`).

Para obtener información sobre el ciclo de vida de una sesión y sobre cómo el controlador QLDB gestiona las sesiones al ejecutar transacciones de datos, consulte Gestión de sesiones con el controlador. Para conocer las prácticas recomendadas para configurar un grupo de sesiones en su aplicación mediante el controlador QLDB, consulte Configuración del objeto QLDBDriver en las recomendaciones de controlador Amazon QLDB.

Verificación de datos en Amazon QLDB

Con Amazon QLDB, puede confiar en que el historial de cambios en los datos de su aplicación es preciso. QLDB utiliza un registro transaccional inmutable, conocido como diario, para el almacenamiento de datos. El diario realiza un seguimiento de cada cambio en los datos confirmados y mantiene un historial de cambios completo y que se pueda verificar con el paso del tiempo.

QLDB utiliza la función hash SHA-256 con un modelo basado en el árbol de Merkle para generar una representación criptográfica de su diario, conocida como resumen. El resumen actúa como una firma única de todo el historial de cambios de sus datos en un momento dado. El resumen se utiliza para comprobar la integridad de las revisiones de los documentos en relación con esa firma.

Temas

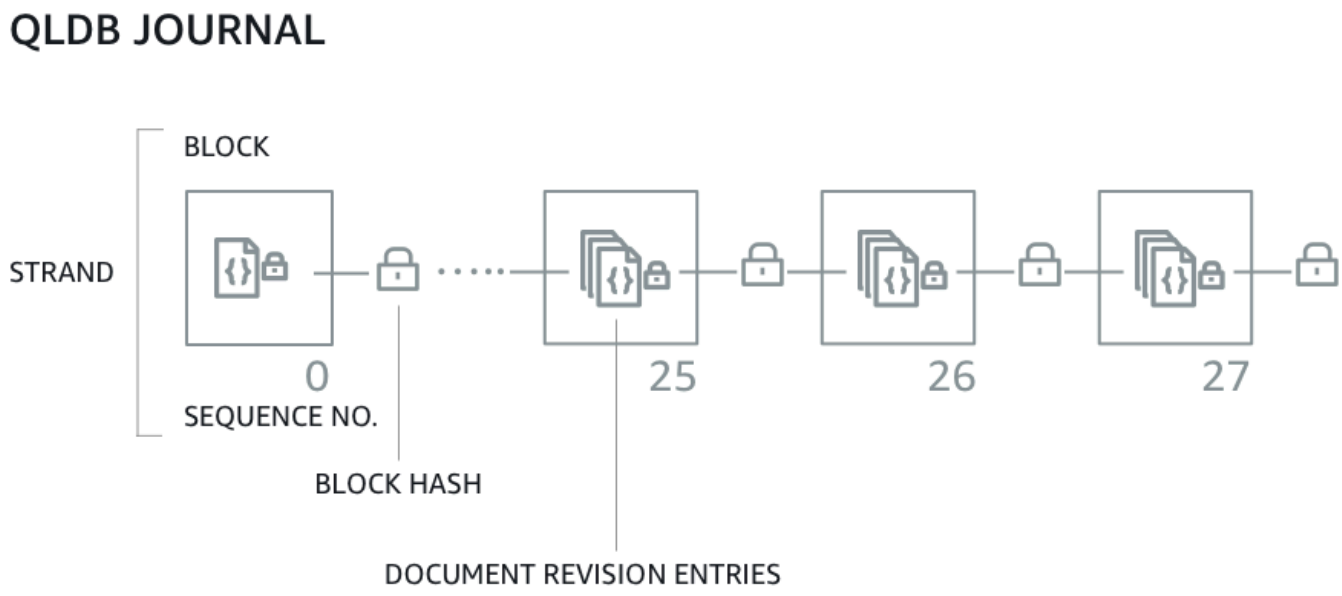
¿Qué tipo de datos se pueden verificar en QLDB?

En QLDB, cada libro mayor tiene exactamente un diario. Un diario puede tener varias cadenas, que son particiones del diario.

> **Note**
>
> Actualmente, QLDB admite diarios con una sola cadena.

Un bloque es un objeto que se asigna a la cadena del diario durante una transacción. Este bloque contiene objetos de entrada, que representan las revisiones de documentos resultantes de la transacción. Puede verificar una revisión individual o un bloque de diario completo en QLDB.

El siguiente diagrama ilustra la estructura de este diario.



El diagrama muestra que las transacciones se registran en el diario como bloques que contienen entradas de revisión de documentos. También muestra que cada bloque está encadenado a los bloques subsiguientes y tiene un número de secuencia para especificar su dirección dentro de la cadena.

Para obtener información sobre el contenido de los datos en un bloque, consulte Contenido del diario en Amazon QLDB.

¿Qué significa integridad de los datos?

La integridad de los datos en QLDB significa que el diario de su libro mayor es, de hecho, inmutable. En otras palabras, sus datos (específicamente, cada revisión de un documento) se encuentran en un estado en el que se cumple lo siguiente:

1. Se encuentran en el mismo lugar de su diario en el que se escribieron por primera vez.
2. No se han modificado de ninguna manera desde que se escribieron.

¿Cómo funciona la verificación?

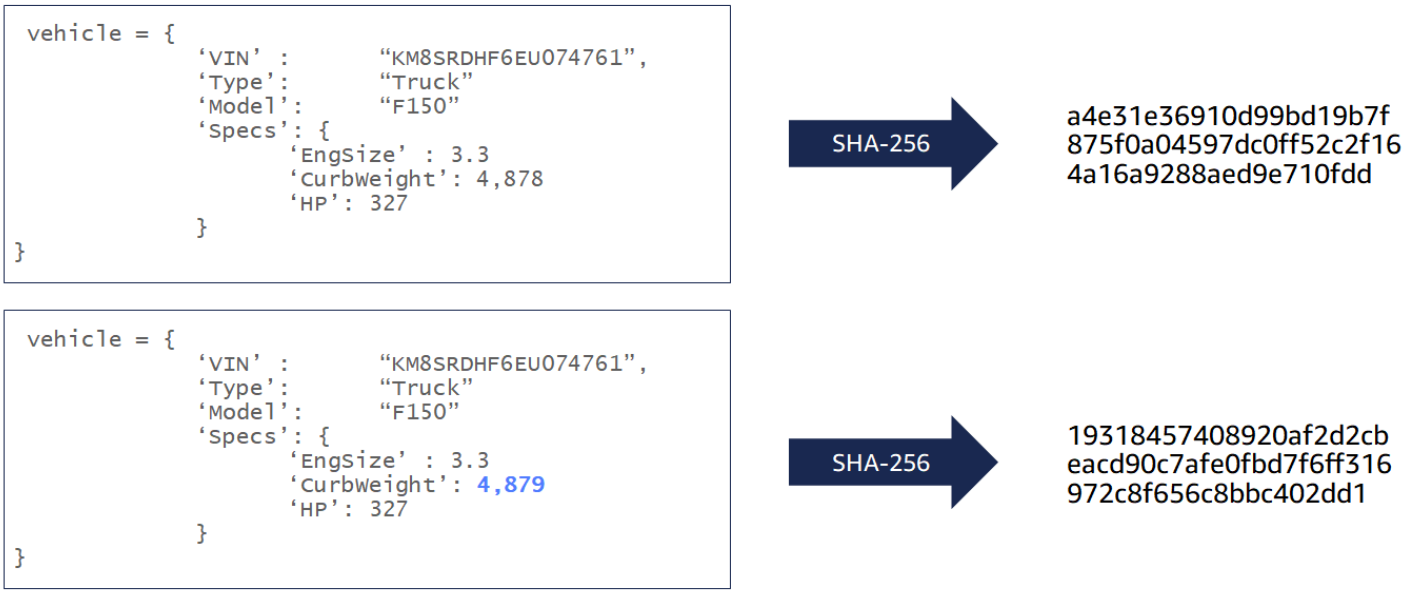
Para entender cómo funciona la verificación en Amazon QLDB, puede dividir el concepto en cuatro componentes básicos.

- Hashing
- Resumir
- Árbol de Merkle
- Prueba

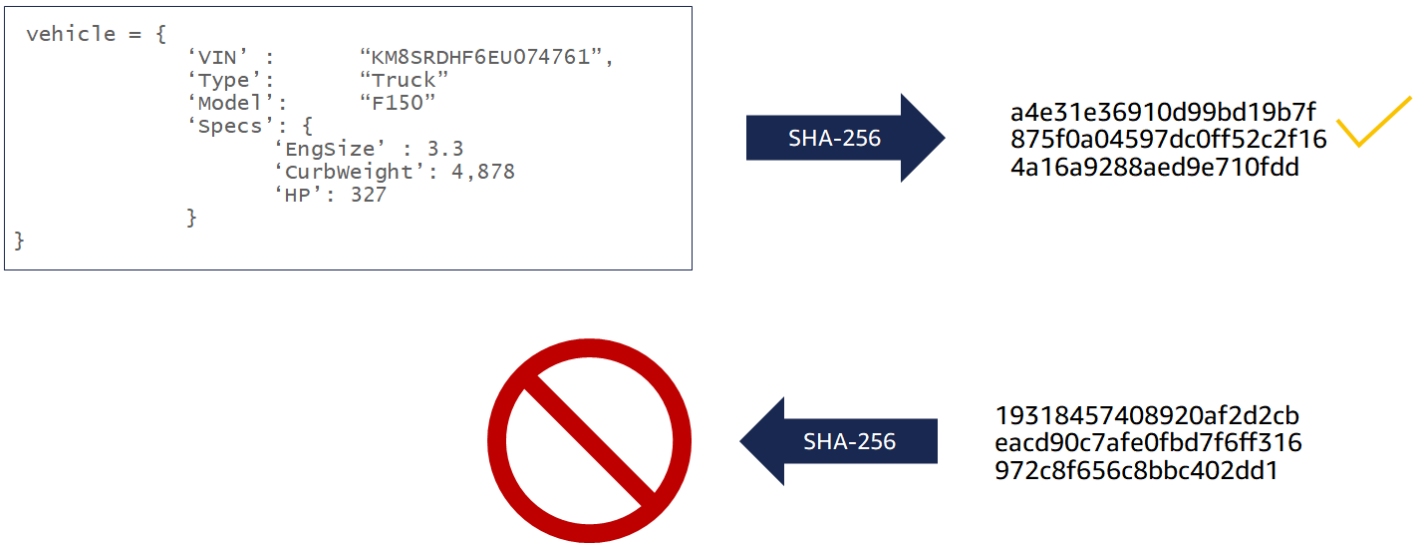
Hashing

QLDB utiliza la función hash criptográfica SHA-256 para crear valores hash de 256 bits. Un hash actúa como una firma única de longitud fija de cualquier cantidad arbitraria de datos de entrada. Si cambia cualquier parte de la entrada (incluso un solo carácter o un bit), el hash de salida cambia por completo.

El siguiente diagrama muestra que la función hash SHA-256 crea valores hash completamente únicos para dos documentos QLDB que difieren solo en un dígito.



La función hash SHA-256 es unidireccional, lo que significa que no es matemáticamente factible calcular la entrada cuando se da una salida. El siguiente diagrama muestra que no es posible calcular el documento QLDB de entrada cuando se le da un valor hash de salida.



Las siguientes entradas de datos se codifican en QLDB con fines de verificación:

- Revisiones del documento
- Instrucciones PartiQL
- Entradas de revisión
- Bloques de diario

Resumir

Un resumen es una representación criptográfica de todo el diario de su libro mayor en un momento dado. Un diario es solo apéndice, y los bloques del diario están secuenciados y encadenados de forma similar a las cadenas de bloques.

Puede solicitar un resumen de un libro mayor en cualquier momento. QLDB genera el resumen y se lo devuelve como un archivo de salida seguro. Luego, use ese resumen para verificar la integridad de las revisiones de los documentos que se realizaron en un momento anterior. Si se vuelven a calcular las divisiones empezando por una revisión y finalizando por el resumen, se demuestra que los datos no se han modificado en el transcurso.

Árbol de Merkle

A medida que aumenta el tamaño del libro mayor, resulta cada vez más ineficiente volver a calcular toda la cadena de hash del diario para su verificación. QLDB utiliza un modelo de árbol de Merkle para abordar esta ineficiencia.

Un árbol de Merkle es una estructura de datos de árbol en la que cada nodo de hoja representa un hash de un bloque de datos. Cada nodo que no es hoja es un hash de sus nodos secundarios. Comúnmente utilizado en las cadenas de bloques, un árbol de Merkle ayuda a verificar de manera eficiente grandes conjuntos de datos con un mecanismo a prueba de auditorías. Para obtener más información sobre los árboles de Merkle, consulte la página de Wikipedia sobre los árboles de Merkle. Para obtener más información sobre las pruebas de auditoría de Merkle y ver un ejemplo de caso de uso, consulte Cómo funcionan las pruebas de registro en el sitio sobre transparencia de certificados.

La implementación QLDB del árbol de Merkle se construye a partir de la cadena hash completa de un diario. En este modelo, los nodos de hoja son el conjunto de todos los hashes de revisión de los documentos individuales. El nodo raíz representa el resumen de todo el diario en un momento dado.

Con una prueba de auditoría de Merkle, puede verificar una revisión consultando solo un pequeño subconjunto del historial de revisiones de su libro mayor. Para ello, recorra el árbol desde un nodo de hoja determinado (revisión) hasta su raíz (resumen). A lo largo de esta ruta transversal, se comprimen de forma recursiva pares de nodos hermanos para calcular su hash principal hasta terminar con el resumen. Este recorrido tiene una complejidad temporal de nodos `log(n)` del árbol.

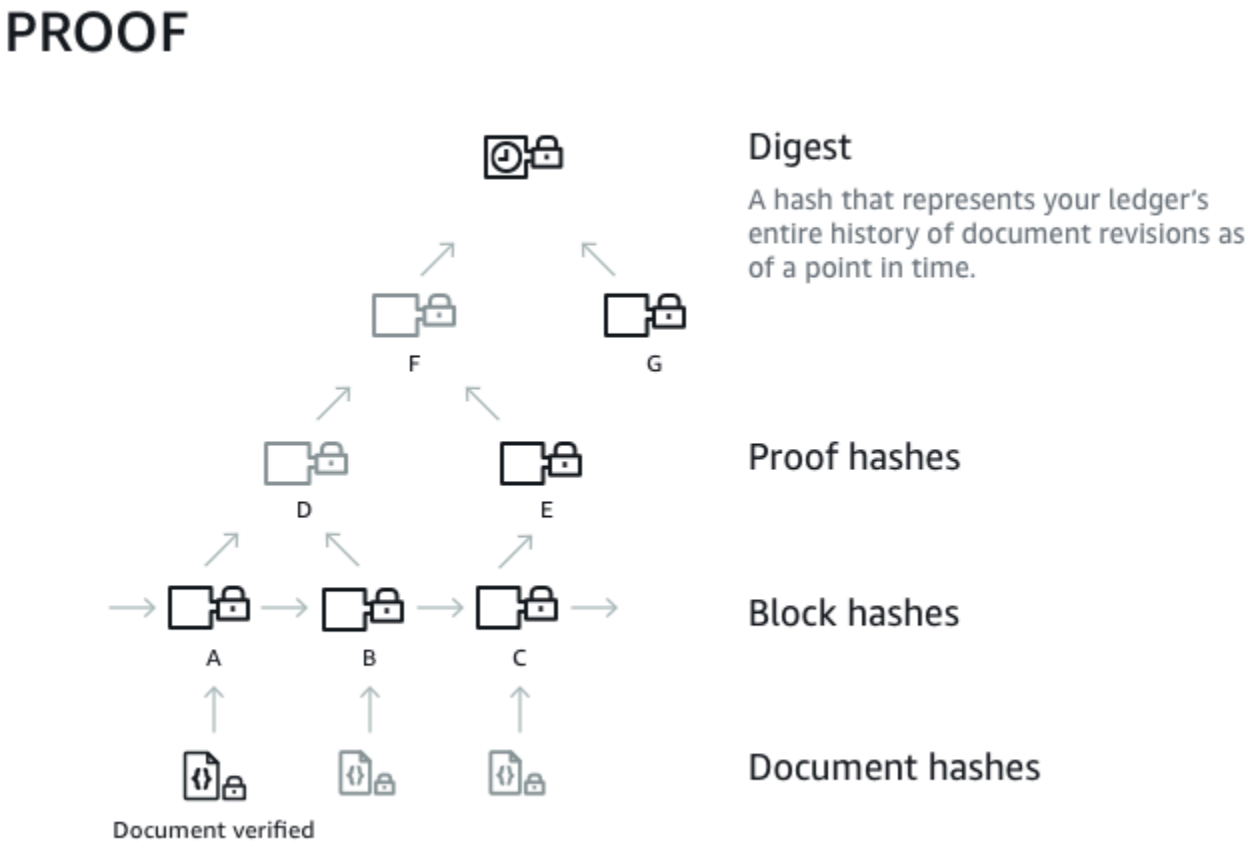
Prueba

Una prueba es la lista ordenada de hashes de nodos que QLDB devuelve para un resumen y una revisión de documentos determinados. Consiste en los hashes que requiere un modelo de árbol de Merkle para encadenar el hash del nodo de hoja dado (una revisión) al hash raíz (el resumen).

Al cambiar los datos confirmados entre una revisión y un resumen, se rompe la cadena de hash del diario y no se puede generar una prueba.

Ejemplo de verificación

El siguiente diagrama ilustra el modelo de árbol de hash de Amazon QLDB. Muestra un conjunto de hashes de bloques que se acumulan hasta el nodo raíz superior, que representa el resumen de una cadena del diario. En un libro mayor con un diario de cadena única, este nodo raíz también es el resumen de todo el libro mayor.



Supongamos que el nodo A es el bloque que contiene la revisión del documento cuyo hash desea verificar. Los siguientes nodos representan la lista ordenada de hashes que QLDB ofrece en su prueba: B, E, G. Estos hashes son necesarios para volver a calcular el resumen a partir del hash A.

Para volver a calcular el resumen, haga lo siguiente:

1. Comience con el hash A y concaténelo con el hash B. Luego, transforme en hash el resultado para calcular D.
2. Use D y E para calcular F.
3. Use F y G para calcular el resumen.

La verificación se realiza correctamente si el resumen recalculado coincide con el valor esperado. Con un hash de revisión y un resumen, no es posible aplicar ingeniería inversa a los códigos hash de una prueba. Por lo tanto, este ejercicio demuestra que su revisión se escribió realmente en la ubicación de este diario en relación con el resumen.

¿Cómo afecta la redacción de los datos a la verificación?

En Amazon QLDB, una instrucción `DELETE` solo elimina un documento de forma lógica al crear una nueva revisión que lo marca como eliminado. QLDB también permite realizar una operación de edición de datos para eliminar permanentemente las revisiones de documentos inactivos del historial de una tabla.

La operación de redacción elimina solo los datos de usuario de la revisión especificada, sin alterar el diario ni los metadatos del documento. Una vez redactada una revisión, los datos de usuario de la revisión (representados por la estructura `data`) se sustituyen por un campo `dataHash` nuevo. El valor de este campo es el hash de Amazon Ion de la estructura `data` eliminada. Para obtener más información y ver un ejemplo de operación de redacción, consulte Editar revisiones de documentos.

Como resultado, el libro mayor mantiene la integridad general de sus datos y sigue siendo verificable criptográficamente mediante las operaciones de la API de verificación existentes. Puedes seguir utilizando estas operaciones de API según lo previsto para solicitar un resumen (GetDigest), solicitar una prueba (GetBlocko GetRevision) y, a continuación, ejecutar tu algoritmo de verificación con los objetos devueltos.

Recalcular un hash de revisión

Si planea verificar una revisión de un documento individual recalculando su hash, debe comprobar condicionalmente si la revisión se ha redactado. Si la revisión se redactó, puede usar el valor hash que se proporciona en el campo `dataHash`. Si no estaba redactada, puede volver a calcular el hash utilizando el campo `data`.

Al realizar esta comprobación condicional, puede identificar las revisiones redactadas y tomar las medidas adecuadas. Por ejemplo, puede registrar los eventos de manipulación de datos con fines de supervisión.

Introducción a las verificaciones

Antes de poder verificar los datos, debe solicitar un resumen a su libro mayor y guardarlo para más adelante. Cualquier revisión de un documento que se efectúe antes del último bloque incluido en el resumen podrá ser cotejada con ese resumen.

A continuación, solicita una prueba a Amazon QLDB para una revisión apta que desee verificar. Con esta prueba, llama a una API del cliente para volver a calcular el resumen, empezando por el hash

de la revisión. Siempre que el resumen guardado anteriormente sea conocido y confiable fuera de QLDB, la integridad del documento quedará demostrada si el hash del resumen recalculado coincide con el hash del resumen guardado.

> **Important**
>
> - Lo que está demostrando específicamente es que la revisión del documento no se modificó entre el momento en que guardó este resumen y el momento en que realizó la verificación. Puede solicitar y guardar un resumen tan pronto como se publique en el diario una revisión que desee verificar más adelante.
> - Como práctica recomendada, sugerimos que solicite resúmenes de forma regular y los guarde aparte del libro mayor. Determine la frecuencia con la que solicita resúmenes en función de la frecuencia con la que realiza las revisiones en su libro mayor.
>
> Para ver una entrada de AWS blog detallada en la que se analiza el valor de la verificación criptográfica en el contexto de un caso de uso realista, consulte Verificación criptográfica en el mundo real con Amazon QLDB.

Para obtener step-by-step guías sobre cómo solicitar un resumen del libro mayor y, a continuación, verificar los datos, consulte lo siguiente:

- Paso 1: solicitar un resumen en QLDB
- Paso 2: verificar los datos en QLDB

Paso 1: solicitar un resumen en QLDB

Amazon QLDB proporciona una API para solicitar un resumen que incluya la sugerencia actual del diario del libro mayor. La sugerencia del diario hace referencia al último bloque confirmado en el momento en que QLDB recibe su solicitud. Puedes usar el AWS Management Console, un AWS SDK o el AWS Command Line Interface (AWS CLI) para obtener un resumen.

Temas

- AWS Management Console
- API DE QLDB

AWS Management Console

Siga estos pasos para restaurar un resumen utilizando la consola de QLDB.

Solicitud de un resumen (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Libros mayores.
3. En la lista de libros mayores, seleccione el nombre del libro mayor para el que desee solicitar un resumen.
4. Seleccione Obtener resumen. El cuadro de diálogo Obtener resumen muestra los siguientes detalles del resumen:

   - Resumen: el valor hash SHA-256 del resumen que ha solicitado.
   - Dirección del tip del resumen: la última ubicación de bloque del diario incluida en el resumen que ha solicitado. Una dirección tiene los dos campos siguientes:
     - `strandId`: el identificador único de la cadena del diario que contiene el bloque.
     - `sequenceNo`: el número de índice que especifica la ubicación del bloque dentro de la cadena.
   - Libro mayor: nombre del libro mayor para el que ha solicitado un resumen.
   - Fecha: fecha y hora en que solicitó el resumen.
5. Revise la información del resumen. A continuación, elija Guardar. Puede conservar el nombre de archivo predeterminado o introducir un nombre nuevo.

   > **Note**
   >
   > Puede que note que los valores de las direcciones hash y sugerencia del resumen cambian incluso cuando no modifica ningún dato del libro mayor. Esto se debe a que la consola recupera el catálogo del sistema del libro mayor cada vez que ejecuta una consulta en el editor PartiQL. Se trata de una transacción de lectura que se registra en el diario y provoca el cambio de la última dirección de bloque.

   Este paso guarda un archivo de texto sin formato con el contenido en formato Amazon Ion. El archivo tiene una extensión de nombre de archivo de `.ion.txt` y contiene toda la información

resumida que aparecía en el cuadro de diálogo anterior. A continuación se muestra un extracto de ejemplo del contenido de un archivo de resumen. El orden de los campos puede variar en función del navegador.

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

6. Guarde este archivo donde pueda acceder a él en el futuro. Más adelante, puede usar este archivo para comparar la revisión de un documento.

> **Important**
>
> La revisión del documento que verifique más adelante debe estar incluida en el resumen que guardó. Es decir, el número de secuencia de la dirección del documento debe ser menor o igual que el número de secuencia de la Dirección del tip del resumen.

API DE QLDB

También puede solicitar un resumen de su libro mayor mediante la API de Amazon QLDB con AWS un SDK o el AWS CLI. La API de QLDB ofrece la siguiente operación para usarla en los programas de aplicación:

- GetDigest— Devuelve el resumen de un libro mayor en el último bloque comprometido del diario. La respuesta incluye un valor hash de 256 bits y una dirección de bloque.

Para obtener información sobre cómo solicitar un resumen mediante el AWS CLI, consulte el comando get-digest en la AWS CLI Referencia de comandos.

Aplicación de muestra

Para ver ejemplos de código Java, consulte el GitHub repositorio amazon-qldb-dmv-sampleaws-samples/ -java. Para obtener instrucciones acerca de cómo descargar e instalar esta aplicación de ejemplo, consulte Instalación de la aplicación de ejemplo Java de Amazon QLDB. Antes de solicitar

un resumen, asegúrese de seguir los pasos del 1 al 3 de Tutorial de Java para crear un libro mayor de muestra y cargarlo con datos de ejemplo.

El código del tutorial de la clase GetDigestproporciona un ejemplo de cómo solicitar un resumen del libro mayor de muestras. `vehicle-registration`

Para verificar la revisión de un documento utilizando el resumen que ha guardado, continúe con Paso 2: verificar los datos en QLDB.

Paso 2: verificar los datos en QLDB

Amazon QLDB ofrece una API para solicitar una prueba de un identificador de documento específico y su bloque asociado. También debe facilitar la dirección de sugerencia de un resumen que haya guardado anteriormente, tal y como se describe en Paso 1: solicitar un resumen en QLDB. Puedes usar el AWS Management Console, un AWS SDK o el AWS CLI para obtener una prueba.

Luego, puede usar la prueba devuelta por QLDB para verificar la revisión del documento con el resumen guardado, mediante una API del cliente. Esto le otorga el control sobre el algoritmo que utiliza para verificar los datos.

Temas

- AWS Management Console
- API DE QLDB

AWS Management Console

En esta sección se describen los pasos para verificar la revisión de un documento con un resumen guardado anteriormente mediante la consola de Amazon QLDB.

Antes de comenzar, asegúrese de que ha realizado los pasos que se detallan en Paso 1: solicitar un resumen en QLDB. La verificación requiere un resumen previamente guardado que incluya la revisión que desea verificar.

Verificación de la revisión de un documento (consola)

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En primer lugar, consulte en el libro mayor para la `id` y la `blockAddress` de la revisión que desee comprobar. Estos campos se incluyen en los metadatos del documento, que puede consultar en la vista confirmada.

El documento `id` es una cadena de identificación única asignada por el sistema. `blockAddress` es una estructura de Ion que especifica la ubicación del bloque en la que se efectuó la revisión.

En el panel de navegación, elija Editor PartiQL.

3. Elija el nombre del libro mayor en el que desee verificar una revisión.
4. En el editor de consultas, introduzca una instrucción `SELECT` en la siguiente sintaxis, y elija Ejecutar.

```
SELECT metadata.id, blockAddress FROM _ql_committed_table_name
WHERE criteria
```

Por ejemplo, la siguiente consulta devuelve un documento de la tabla `VehicleRegistration` del ejemplo de libro mayor creado en Introducción a la consola de Amazon QLDB.

```
SELECT r.metadata.id, r.blockAddress FROM _ql_committed_VehicleRegistration AS r
WHERE r.data.VIN = 'KM8SRDHF6EU074761'
```

5. Copie y guarde los valores `id` y `blockAddress` que devuelve la consulta. Asegúrese de omitir las comillas dobles del campo `id`. En Amazon Ion, los tipos de datos de cadena se delimitan con comillas dobles. Por ejemplo, debe copiar únicamente el texto alfanumérico del siguiente fragmento.

"*LtMNJYNjSwzBLgf7sLifrG*"

6. Ahora que ha seleccionado una revisión del documento, puede iniciar el proceso de verificación.

En el panel de navegación izquierdo, elija Verificaciones.

7. En el formulario Verificar documento, en Especifique el documento que desea verificar, introduzca los siguientes parámetros de entrada:

- Libro mayor: el libro mayor en el que desee verificar una revisión.
- Dirección del bloque: el valor `blockAddress` devuelto por la consulta en el paso 4.
- ID del documento: el valor `id` devuelto por la consulta en el paso 4.

8. En Especificar el resumen que se va a usar para la verificación, seleccione el resumen que guardó anteriormente; para ello, seleccione Elegir resumen. Si el archivo es válido, se rellenan automáticamente todos los campos de resumen de la consola. O bien, puede copiar y pegar manualmente los siguientes valores directamente desde el archivo de resumen:

- Resumen: el valor `digest` del archivo de resumen.
- Dirección del tip del resumen: el valor `digestTipAddress` del archivo de resumen.

9. Revise los parámetros de entrada del documento y del resumen y, a continuación, seleccione Verificar.

   La consola automatiza dos pasos:

   a. Solicite una prueba a QLDB para el documento especificado.

   b. Utilice la prueba devuelta por QLDB para llamar a una API del cliente que verifica la revisión de su documento comparándola con el resumen proporcionado. Para examinar este algoritmo de verificación, consulte la siguiente sección API DE QLDB para descargar el ejemplo de código.

   La consola muestra los resultados de su solicitud en la tarjeta de Resultados de la verificación. Para obtener más información, consulte Resultados de verificación.

API DE QLDB

También puede verificar la revisión de un documento mediante la API de Amazon QLDB con AWS un SDK o el AWS CLI. La API de QLDB ofrece las siguientes operaciones para usarlas en los programas de aplicación:

- `GetDigest`: devuelve el resumen de un libro mayor en el último bloque comprometido del diario. La respuesta incluye un valor hash de 256 bits y una dirección de bloque.
- `GetBlock`: devuelve un objeto de bloque en una dirección específica de un diario. También devuelve una prueba del bloque especificado para su verificación, si `DigestTipAddress` se proporciona.
- `GetRevision`: devuelve un objeto de datos de revisión para un identificador de documento y una dirección de bloque específicos. También devuelve una prueba de la revisión especificada para su verificación, si se proporciona `DigestTipAddress`.

Para obtener una descripción completa de estas operaciones de API, consulte la Referencia de la API de Amazon QLDB.

Para obtener información sobre la verificación de datos mediante el AWS CLI, consulte la Referencia de AWS CLI comandos.

Aplicación de muestra

Para ver ejemplos de código Java, consulte el GitHub repositorio amazon-qldb-dmv-sampleaws-samples/ -java. Para obtener instrucciones acerca de cómo descargar e instalar esta aplicación de ejemplo, consulte Instalación de la aplicación de ejemplo Java de Amazon QLDB. Antes de realizar una verificación, siga los pasos del 1 al 3 de Tutorial de Java para crear un libro mayor de muestra y cargarlo con datos de ejemplo.

El código del tutorial de la clase GetRevisionproporciona un ejemplo de cómo solicitar una prueba para la revisión de un documento y, a continuación, verificar esa revisión. Esta clase ejecuta los siguientes pasos:

1. Solicite un nuevo resumen del libro mayor de muestras `vehicle-registration`.
2. Solicite una prueba de revisión de un documento de muestra de la tabla `VehicleRegistration` del libro mayor `vehicle-registration`.
3. Verifique la revisión de la muestra utilizando el resumen y la prueba devueltos.

Resultados de verificación

En esta sección se describen los resultados devueltos por una solicitud de verificación de datos de Amazon QLDB en la AWS Management Console. Para ver pasos detallados acerca de cómo enviar una solicitud de verificación, consulte Paso 2: verificar los datos en QLDB.

En la página de Verificación de la consola de QLDB, los resultados de su solicitud se muestran en la tarjeta de Resultados de la verificación. La pestaña Prueba muestra el contenido de la prueba devuelta por QLDB para la revisión y el resumen del documento especificados. Contiene los datos siguientes:

- Hash de revisión: el valor SHA-256 que representa de forma única la revisión del documento que está verificando.
- Hashes de prueba: la lista ordenada de hashes proporcionada por QLDB que se utilizan para volver a calcular el resumen especificado. La consola comienza con el Hash de revisión y lo combina secuencialmente con cada hash de prueba hasta que termina con un resumen recalculado.

La lista está contraída de forma predeterminada, por lo que puede expandirla para mostrar los valores del hash. Si lo desea, puede probar los cálculos de hash usted mismo siguiendo los pasos que se describen en Uso de una prueba para volver a calcular el resumen.

- Resumen calculado: el hash resultante de la serie de cálculos de hash que se realizaron con el hash de revisión. Si este valor coincide con el Resumen guardado anteriormente, la verificación se ha realizado correctamente.

La pestaña Bloquear muestra el contenido del bloque que contiene la revisión que estás verificando. Contiene los datos siguientes:

- ID de transacción: el identificador único de la transacción que confirmó este bloqueo.
- Hora de la transacción: la marca temporal en la que este bloque se asignó a la cadena.
- Hash de bloque: el valor SHA-256 que representa de forma única este bloque y todo su contenido.
- Dirección de bloque: la ubicación en el diario del libro mayor en la que se consignó este bloqueo. Una dirección tiene los dos campos siguientes:
  - ID de cadena: el identificador único de la cadena del diario que contiene el bloque.
  - Número de secuencia: el número de índice que especifica la ubicación del bloque dentro de la cadena.
- Declaraciones: las sentencias PartiQL que se ejecutaron para confirmar las entradas de este bloque.

> **Note**
>
> Si ejecuta sentencias parametrizadas mediante programación, se graban en los bloques de su diario con parámetros de enlace en lugar de con datos literales. Por ejemplo, es posible que vea la siguiente instrucción en un bloque de diario, donde el signo de interrogación (?) es un marcador de posición variable para el contenido del documento.
>
> ```
> INSERT INTO Vehicle ?
> ```

- Entradas del documento: las revisiones del documento que se realizaron en este bloque.

Si su solicitud no pudo verificar la revisión del documento, consulte Errores comunes de verificación para obtener información sobre las posibles causas.

Uso de una prueba para volver a calcular el resumen

Una vez que QLDB devuelva una prueba de su solicitud de verificación de documentos, puede intentar realizar los cálculos de hash usted mismo. En esta sección se describen los pasos básicos para volver a calcular su resumen utilizando la prueba que se proporciona.

Primero, empareje su hash de revisión con el primer hash de la lista de hashes de prueba. A continuación, proceda del modo siguiente.

1. Ordene los dos hashes. Compare los hashes por sus valores de bytes firmados en orden little-endian.
2. Concatene los dos hashes en orden.
3. Aplique un hash al par concatenado con un generador de hash SHA-256.
4. Empareje el nuevo hash con el siguiente hash de la prueba y repita los pasos del 1 al 3. Después de procesar el último hash de prueba, el nuevo hash es el resumen recalculado.

Si el resumen recalculado coincide con el resumen guardado anteriormente, el documento se ha verificado correctamente.

Para ver un step-by-step tutorial con ejemplos de código que muestran estos pasos de verificación, continúe Tutorial: Verifying data using an AWS SDK con.

Tutorial: Verifying data using an AWS SDK

En este tutorial, verificará un hash de revisión de documento y un hash de bloque de diario en un libro mayor de Amazon QLDB mediante la API de QLDB a través de un SDK. AWS También utiliza el controlador de QLDB para consultar la revisión del documento.

Considere un ejemplo en el que tiene una revisión de un documento que contiene datos de un vehículo con un número de identificación del vehículo (VIN) de `KM8SRDHF6EU074761`. La revisión del documento se encuentra en una tabla `VehicleRegistration` que se encuentra en un libro mayor denominado `vehicle-registration`. Supongamos que desea comprobar la integridad tanto de la revisión del documento de este vehículo como del bloque de diario que contiene la revisión.

> **Note**
>
> Para ver una entrada de AWS blog detallada en la que se analiza el valor de la verificación criptográfica en el contexto de un caso de uso realista, consulte Verificación criptográfica en el mundo real con Amazon QLDB.

Temas

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Configure el controlador de QLDB para el idioma de su elección completando los requisitos previos correspondientes que se indican en Introducción al controlador Amazon QLDB. Esto incluye registrarse AWS, conceder acceso programático para el desarrollo y configurar su entorno de desarrollo.
2. Siga los pasos 1 y 2 de Introducción a la consola de Amazon QLDB para crear un libro mayor denominado `vehicle-registration` y cárguelo con datos de ejemplo predefinidos.

A continuación, revise los siguientes pasos para aprender cómo funciona la verificación y, a continuación, ejecute el ejemplo de código completo de principio a fin.

Paso 1: Solicitar un resumen

Antes de poder verificar los datos, primero debe solicitar un resumen a su libro mayor `vehicle-registration` para usarlo más adelante.

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')
```

Paso 2: Consultar la revisión del documento

Utilice el controlador de QLDB para consultar las direcciones de bloque, los hashes y los ID de documentos asociados al VIN `KM8SRDHF6EU074761`.

Java

```
// Retrieve info for the given vin's document revisions
Result result = driver.execute(txn -> {
    final String query = String.format("SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_%s WHERE data.VIN = '%s'", tableName, vin);
    return txn.execute(query);
});
```

.NET

```
// Retrieve info for the given vin's document revisions
```

```
var result = driver.Execute(txn => {
    string query = $"SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_{tableName} WHERE data.VIN = '{vin}'";
    return txn.Execute(query);
});
```

Go

```
// Retrieve info for the given vin's document revisions
result, err := driver.Execute(context.Background(), func(txn qldbdriver.Transaction)
 (interface{}, error) {
    statement := fmt.Sprintf(
            "SELECT blockAddress, hash, metadata.id FROM _ql_committed_%s WHERE
 data.VIN = '%s'",
            tableName,
            vin)
    result, err := txn.Execute(statement)
    if err != nil {
        return nil, err
    }

    results := make([]map[string]interface{}, 0)

    // Convert the result set into a map
    for result.Next(txn) {
        var doc map[string]interface{}
        err := ion.Unmarshal(result.GetCurrentData(), &doc)
        if err != nil {
            return nil, err
        }
        results = append(results, doc)
    }
    return results, nil
})
if err != nil {
    panic(err)
}
resultSlice := result.([]map[string]interface{})
```

Node.js

```
const result: dom.Value[] = await driver.executeLambda(async (txn:
 TransactionExecutor): Promise<dom.Value[]> => {
```

```
    const query: string = `SELECT blockAddress, hash, metadata.id FROM
 _ql_committed_${tableName} WHERE data.VIN = '${vin}'`;
    const queryResult: Result = await txn.execute(query);
    return queryResult.getResultList();
});
```

Python

```
def query_doc_revision(txn):
    query = "SELECT blockAddress, hash, metadata.id FROM _ql_committed_{} WHERE
 data.VIN = '{}'".format(table_name, vin)
    return txn.execute_statement(query)

# Retrieve info for the given vin's document revisions
result = qldb_driver.execute_lambda(query_doc_revision)
```

Paso 3: Solicitar una prueba para la revisión

Revise los resultados de la consulta y utilice cada dirección de bloque e ID de documento junto con el nombre del libro mayor para enviar una solicitud `GetRevision`. Para obtener una prueba de la revisión, también debe proporcionar la dirección del tip que figura en el resumen guardado anteriormente. Esta operación de API devuelve un objeto que incluye la revisión del documento y la prueba de la revisión.

Para obtener más información acerca de la estructura de la revisión y sus contenidos, consulte Consulta de los metadatos del documento.

Java

```
for (IonValue ionValue : result) {
    IonStruct ionStruct = (IonStruct)ionValue;

    // Get the requested fields
    IonValue blockAddress = ionStruct.get("blockAddress");
    IonBlob hash = (IonBlob)ionStruct.get("hash");
    String metadataId = ((IonString)ionStruct.get("id")).stringValue();

    System.out.printf("Verifying document revision for id '%s'%n", metadataId);

    String blockAddressText = blockAddress.toString();
```

```
    // Submit a request for the revision
    GetRevisionRequest revisionRequest = new GetRevisionRequest()
            .withName(ledgerName)
            .withBlockAddress(new ValueHolder().withIonText(blockAddressText))
            .withDocumentId(metadataId)
            .withDigestTipAddress(digestResult.getDigestTipAddress());

    // Get a result back
    GetRevisionResult revisionResult = client.getRevision(revisionRequest);

    ...
}
```

.NET

```
foreach (IIonValue ionValue in result)
{
    IIonStruct ionStruct = ionValue;

    // Get the requested fields
    IIonValue blockAddress = ionStruct.GetField("blockAddress");
    IIonBlob hash = ionStruct.GetField("hash");
    String metadataId = ionStruct.GetField("id").StringValue;

    Console.WriteLine($"Verifying document revision for id '{metadataId}'");

    // Use an Ion Reader to convert block address to text
    IIonReader reader = IonReaderBuilder.Build(blockAddress);
    StringWriter sw = new StringWriter();
    IIonWriter textWriter = IonTextWriterBuilder.Build(sw);
    textWriter.WriteValues(reader);
    string blockAddressText = sw.ToString();

    // Submit a request for the revision
    GetRevisionRequest revisionRequest = new GetRevisionRequest
    {
        Name = ledgerName,
        BlockAddress = new ValueHolder
        {
            IonText = blockAddressText
        },
        DocumentId = metadataId,
        DigestTipAddress = getDigestResponse.DigestTipAddress
```

```
    };

    // Get a response back
    GetRevisionResponse revisionResponse =
 client.GetRevisionAsync(revisionRequest).Result;

    ...
}
```

Go

```
for _, value := range resultSlice {
    // Get the requested fields
    ionBlockAddress, err := ion.MarshalText(value["blockAddress"])
    if err != nil {
        panic(err)
    }
    blockAddress := string(ionBlockAddress)
    metadataId := value["id"].(string)
    documentHash := value["hash"].([]byte)

    fmt.Printf("Verifying document revision for id '%s'\n", metadataId)

    // Submit a request for the revision
    revisionInput := qldb.GetRevisionInput{
        BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
        DigestTipAddress: digestOutput.DigestTipAddress,
        DocumentId:       &metadataId,
        Name:             &currentLedgerName,
    }

    // Get a result back
    revisionOutput, err := client.GetRevision(&revisionInput)
    if err != nil {
        panic(err)
    }

    ...
}
```

Node.js

```
for (let value of result) {
```

```
    // Get the requested fields
    const blockAddress: dom.Value = value.get("blockAddress");
    const hash: dom.Value = value.get("hash");
    const metadataId: string = value.get("id").stringValue();

    console.log(`Verifying document revision for id '${metadataId}'`);

    // Submit a request for the revision
    const revisionRequest: GetRevisionRequest = {
        Name: ledgerName,
        BlockAddress: {
            IonText: dumpText(blockAddress)
        },
        DocumentId: metadataId,
        DigestTipAddress: getDigestResponse.DigestTipAddress
    };

    // Get a response back
    const revisionResponse: GetRevisionResponse = await
 qldbClient.getRevision(revisionRequest).promise();

    ...
}
```

Python

```
for value in result:
    # Get the requested fields
    block_address = value['blockAddress']
    document_hash = value['hash']
    metadata_id = value['id']

    print("Verifying document revision for id '{}'".format(metadata_id))

    # Submit a request for the revision and get a result back
    proof_response = qldb_client.get_revision(Name=ledger_name,

 BlockAddress=block_address_to_dictionary(block_address),
                                          DocumentId=metadata_id,
                                          DigestTipAddress=digest_tip_address)
```

A continuación, recupere la prueba de la revisión solicitada.

La API QLDB devuelve la prueba como una representación en cadena de la lista ordenada de hashes de nodos. Para convertir esta cadena en una lista de la representación binaria de los hashes de los nodos, puede utilizar un lector Ion de la biblioteca Amazon Ion. Para obtener más información sobre el uso de la biblioteca Ion, consulte Amazon Ion Cookbook.

Java

En este ejemplo, se utiliza `IonReader` para realizar la conversión binaria.

```
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

En este ejemplo, se utiliza `IonLoader` para cargar la prueba en un datagrama de Ion.

```
string proofText = revisionResponse.Proof.IonText;
IIonDatagram proofValue = IonLoader.Default.Load(proofText);
```

Go

En este ejemplo, se utiliza un lector de Ion para convertir la prueba en binaria y recorrer en iteración la lista de hashes de nodos de la prueba.

```
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

En este ejemplo, se utiliza la característica `load` para realizar la conversión binaria.

```
let proofValue: dom.Value = load(revisionResponse.Proof.IonText);
```

Python

En este ejemplo, se utiliza la característica `loads` para realizar la conversión binaria.

```
proof_text = proof_response.get('Proof').get('IonText')
proof_hashes = loads(proof_text)
```

Paso 4: Volver a calcular el resumen de la revisión

Use la lista de hashes de la prueba para volver a calcular el resumen, empezando por el hash de la revisión. Siempre que el resumen guardado anteriormente sea conocido y confiable fuera de QLDB, la integridad de la revisión del documento queda demostrada si el hash del resumen recalculado coincide con el hash del resumen guardado.

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))
```

Paso 5: Solicitar una prueba para el bloque de diario

A continuación, se verifica el bloque de diario que contiene la revisión del documento.

Use la dirección del bloque y la dirección del tip del resumen que guardó en el paso 1 para enviar una solicitud `GetBlock`. Al igual que en la solicitud `GetRevision` del paso 2, debe volver a proporcionar la dirección del tip que aparece en el resumen guardado para obtener una prueba del bloqueo. Esta operación de API devuelve un objeto que incluye el bloque y la prueba del bloque.

Para obtener más información acerca de la estructura del bloque y su contenido, consulte Contenido del diario en Amazon QLDB.

Java

```
// Submit a request for the block
GetBlockRequest getBlockRequest = new GetBlockRequest()
        .withName(ledgerName)
```

```
        .withBlockAddress(new ValueHolder().withIonText(blockAddressText))
        .withDigestTipAddress(digestResult.getDigestTipAddress());

// Get a result back
GetBlockResult getBlockResult = client.getBlock(getBlockRequest);
```

.NET

```
// Submit a request for the block
GetBlockRequest getBlockRequest = new GetBlockRequest
{
    Name = ledgerName,
    BlockAddress = new ValueHolder
    {
        IonText = blockAddressText
    },
    DigestTipAddress = getDigestResponse.DigestTipAddress
};

// Get a response back
GetBlockResponse getBlockResponse = client.GetBlockAsync(getBlockRequest).Result;
```

Go

```
// Submit a request for the block
blockInput := qldb.GetBlockInput{
    Name:             &currentLedgerName,
    BlockAddress:     &qldb.ValueHolder{IonText: &blockAddress},
    DigestTipAddress: digestOutput.DigestTipAddress,
}

// Get a result back
blockOutput, err := client.GetBlock(&blockInput)
if err != nil {
    panic(err)
}
```

Node.js

```
// Submit a request for the block
const getBlockRequest: GetBlockRequest = {
    Name: ledgerName,
    BlockAddress: {
```

```
        IonText: dumpText(blockAddress)
    },
    DigestTipAddress: getDigestResponse.DigestTipAddress
};

// Get a response back
const getBlockResponse: GetBlockResponse = await
 qldbClient.getBlock(getBlockRequest).promise();
```

Python

```
def block_address_to_dictionary(ion_dict):
"""
Convert a block address from IonPyDict into a dictionary.
Shape of the dictionary must be: {'IonText': "{strandId: <"strandId">, sequenceNo:
 <sequenceNo>}"}

:type ion_dict: :py:class:`amazon.ion.simple_types.IonPyDict`/str
:param ion_dict: The block address value to convert.

:rtype: dict
:return: The converted dict.
"""
block_address = {'IonText': {}}
if not isinstance(ion_dict, str):
    py_dict = '{{strandId: "{}", sequenceNo:{}}}'.format(ion_dict['strandId'],
 ion_dict['sequenceNo'])
    ion_dict = py_dict
block_address['IonText'] = ion_dict
return block_address

# Submit a request for the block and get a result back
block_response = qldb_client.get_block(Name=ledger_name,
 BlockAddress=block_address_to_dictionary(block_address),
                                        DigestTipAddress=digest_tip_address)
```

A continuación, recupere el hash del bloque y la prueba del resultado.

Java

En este ejemplo, se utiliza `IonLoader` para cargar el objeto de bloque en un contenedor `IonDatagram`.

```
String blockText = getBlockResult.getBlock().getIonText();

IonDatagram datagram = SYSTEM.getLoader().load(blockText);
ionStruct = (IonStruct)datagram.get(0);

final byte[] blockHash = ((IonBlob)ionStruct.get("blockHash")).getBytes();
```

También se utiliza `IonLoader` para cargar la prueba en un `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

En este ejemplo, se suele `IonLoader` cargar el bloque y la prueba en un datagrama de Ion para cada uno.

```
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

En este ejemplo, se utiliza un lector de Ion para convertir la prueba en binaria y recorrer en iteración la lista de hashes de nodos de la prueba.

```
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

En este ejemplo, se utiliza la característica `load` para realizar la conversión binaria del bloque y de la prueba.

```
const blockValue: dom.Value = load(getBlockResponse.Block.IonText)
let blockHash: Uint8Array = blockValue.get("blockHash").uInt8ArrayValue();

proofValue = load(getBlockResponse.Proof.IonText);
```

Python

En este ejemplo, se utiliza la característica `loads` para realizar la conversión binaria del bloque y de la prueba.

```
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)
```

Paso 6: Volver a calcular el resumen del bloque

Use la lista de hashes de la prueba para volver a calcular el resumen, empezando por el hash del bloque. Siempre que el resumen guardado anteriormente sea conocido y confiable fuera de QLDB, la integridad del bloque queda demostrada si el hash del resumen recalculado coincide con el hash del resumen guardado.

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))
```

Los ejemplos de código anteriores utilizan la función `dot` siguiente para volver a calcular el resumen. Esta función toma una entrada de dos hashes, los ordena, los concatena y, a continuación, devuelve el hash a la matriz concatenada.

Java

```
/**
 * Takes two hashes, sorts them, concatenates them, and then returns the
 * hash of the concatenated array.
 *
 * @param h1
 *              Byte array containing one of the hashes to compare.
 * @param h2
 *              Byte array containing one of the hashes to compare.
 * @return the concatenated array of hashes.
 */
public static byte[] dot(final byte[] h1, final byte[] h2) {
    if (h1.length != HASH_LENGTH || h2.length != HASH_LENGTH) {
        throw new IllegalArgumentException("Invalid hash.");
    }

    int byteEqual = 0;
    for (int i = h1.length - 1; i >= 0; i--) {
        byteEqual = Byte.compare(h1[i], h2[i]);
        if (byteEqual != 0) {
            break;
        }
    }

    byte[] concatenated = new byte[h1.length + h2.length];
    if (byteEqual < 0) {
        System.arraycopy(h1, 0, concatenated, 0, h1.length);
        System.arraycopy(h2, 0, concatenated, h1.length, h2.length);
    } else {
        System.arraycopy(h2, 0, concatenated, 0, h2.length);
```

```
        System.arraycopy(h1, 0, concatenated, h2.length, h1.length);
    }

    MessageDigest messageDigest;
    try {
        messageDigest = MessageDigest.getInstance("SHA-256");
    } catch (NoSuchAlgorithmException e) {
        throw new IllegalStateException("SHA-256 message digest is unavailable", e);
    }

    messageDigest.update(concatenated);
    return messageDigest.digest();
}
```

.NET

```
/// <summary>
/// Takes two hashes, sorts them, concatenates them, and then returns the
/// hash of the concatenated array.
/// </summary>
/// <param name="h1">Byte array containing one of the hashes to compare.</param>
/// <param name="h2">Byte array containing one of the hashes to compare.</param>
/// <returns>The concatenated array of hashes.</returns>
private static byte[] Dot(byte[] h1, byte[] h2)
{
    if (h1.Length == 0)
    {
        return h2;
    }

    if (h2.Length == 0)
    {
        return h1;
    }

    HashAlgorithm hashAlgorithm = HashAlgorithm.Create("SHA256");
    HashComparer comparer = new HashComparer();
    if (comparer.Compare(h1, h2) < 0)
    {
        return hashAlgorithm.ComputeHash(h1.Concat(h2).ToArray());
    }
    else
    {
```

```
        return hashAlgorithm.ComputeHash(h2.Concat(h1).ToArray());
    }
}

private class HashComparer : IComparer<byte[]>
{
    private static readonly int HASH_LENGTH = 32;

    public int Compare(byte[] h1, byte[] h2)
    {
        if (h1.Length != HASH_LENGTH || h2.Length != HASH_LENGTH)
        {
            throw new ArgumentException("Invalid hash");
        }

        for (var i = h1.Length - 1; i >= 0; i--)
        {
            var byteEqual = (sbyte)h1[i] - (sbyte)h2[i];
            if (byteEqual != 0)
            {
                return byteEqual;
            }
        }

        return 0;
    }
}
```

Go

```
// Takes two hashes, sorts them, concatenates them, and then returns the hash of the
 concatenated array.
func dot(h1, h2 []byte) ([]byte, error) {
    compare, err := hashComparator(h1, h2)
    if err != nil {
        return nil, err
    }

    var concatenated []byte
    if compare < 0 {
        concatenated = append(h1, h2...)
    } else {
        concatenated = append(h2, h1...)
```

```
    }

    newHash := sha256.Sum256(concatenated)
    return newHash[:], nil
}

func hashComparator(h1 []byte, h2 []byte) (int16, error) {
    if len(h1) != hashLength || len(h2) != hashLength {
        return 0, errors.New("invalid hash")
    }
    for i := range h1 {
        // Reverse index for little endianness
        index := hashLength - 1 - i

        // Handle byte being unsigned and overflow
        h1Int := int16(h1[index])
        h2Int := int16(h2[index])
        if h1Int > 127 {
            h1Int = 0 - (256 - h1Int)
        }
        if h2Int > 127 {
            h2Int = 0 - (256 - h2Int)
        }

        difference := h1Int - h2Int
        if difference != 0 {
            return difference, nil
        }
    }
    return 0, nil
}
```

Node.js

```
/**
 * Takes two hashes, sorts them, concatenates them, and calculates a digest based on
 the concatenated hash.
 * @param h1 Byte array containing one of the hashes to compare.
 * @param h2 Byte array containing one of the hashes to compare.
 * @returns The digest calculated from the concatenated hash values.
 */
function dot(h1: Uint8Array, h2: Uint8Array): Uint8Array {
    if (h1.length === 0) {
```

```
        return h2;
    }
    if (h2.length === 0) {
        return h1;
    }

    const newHashLib = createHash("sha256");

    let concatenated: Uint8Array;
    if (hashComparator(h1, h2) < 0) {
        concatenated = concatenate(h1, h2);
    } else {
        concatenated = concatenate(h2, h1);
    }
    newHashLib.update(concatenated);
    return newHashLib.digest();
}

/**
 * Compares two hashes by their **signed** byte values in little-endian order.
 * @param hash1 The hash value to compare.
 * @param hash2 The hash value to compare.
 * @returns Zero if the hash values are equal, otherwise return the difference of
 the first pair of non-matching
 *          bytes.
 * @throws RangeError When the hash is not the correct hash size.
 */
function hashComparator(hash1: Uint8Array, hash2: Uint8Array): number {
    if (hash1.length !== HASH_SIZE || hash2.length !== HASH_SIZE) {
        throw new RangeError("Invalid hash.");
    }
    for (let i = hash1.length-1; i >= 0; i--) {
        const difference: number = (hash1[i]<<24 >>24) - (hash2[i]<<24 >>24);
        if (difference !== 0) {
            return difference;
        }
    }
    return 0;
}

/**
 * Helper method that concatenates two Uint8Array.
 * @param arrays List of arrays to concatenate, in the order provided.
 * @returns The concatenated array.
```

```
 */
function concatenate(...arrays: Uint8Array[]): Uint8Array {
    let totalLength = 0;
    for (const arr of arrays) {
        totalLength += arr.length;
    }
    const result = new Uint8Array(totalLength);
    let offset = 0;
    for (const arr of arrays) {
        result.set(arr, offset);
        offset += arr.length;
    }
    return result;
}

/**
 * Helper method that checks for equality between two Uint8Array.
 * @param expected Byte array containing one of the hashes to compare.
 * @param actual Byte array containing one of the hashes to compare.
 * @returns Boolean indicating equality between the two Uint8Array.
 */
function isEqual(expected: Uint8Array, actual: Uint8Array): boolean {
    if (expected === actual) return true;
    if (expected == null || actual == null) return false;
    if (expected.length !== actual.length) return false;

    for (let i = 0; i < expected.length; i++) {
        if (expected[i] !== actual[i]) {
            return false;
        }
    }
    return true;
}
```

Python

```
def dot(hash1, hash2):
    """
    Takes two hashes, sorts them, concatenates them, and then returns the
    hash of the concatenated array.

    :type hash1: bytes
    :param hash1: The hash value to compare.
```

```
    :type hash2: bytes
    :param hash2: The hash value to compare.

    :rtype: bytes
    :return: The new hash value generated from concatenated hash values.
    """
    if len(hash1) != hash_length or len(hash2) != hash_length:
        raise ValueError('Illegal hash.')

    hash_array1 = array('b', hash1)
    hash_array2 = array('b', hash2)

    difference = 0
    for i in range(len(hash_array1) - 1, -1, -1):
        difference = hash_array1[i] - hash_array2[i]
        if difference != 0:
            break

    if difference < 0:
        concatenated = hash1 + hash2
    else:
        concatenated = hash2 + hash1

    new_hash_lib = sha256()
    new_hash_lib.update(concatenated)
    new_digest = new_hash_lib.digest()
    return new_digest
```

Ejecute el ejemplo de código completo

Ejecute el ejemplo de código completo de la siguiente manera para realizar todos los pasos anteriores de principio a fin.

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))
```

Errores comunes de verificación

En esta sección se describen los errores de tiempo de ejecución que genera Amazon QLDB para las solicitudes de verificación.

La siguiente es una lista de excepciones comunes devueltas por el servicio. Cada excepción incluye el mensaje de error específico, seguido de las operaciones de la API que pueden generarlo, una breve descripción y sugerencias de posibles soluciones.

IllegalArgumentException

Mensaje: The provided Ion value is not valid and cannot be parsed.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

Asegúrese de proporcionar un valor de Amazon Ion válido antes de volver a intentar la solicitud.

IllegalArgumentException

Mensaje: The provided block address is not valid.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

Asegúrese de proporcionar una dirección de bloque válida antes de volver a intentar la solicitud. Una dirección de bloque es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

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

IllegalArgumentException

Mensaje: The sequence number of the provided digest tip address is beyond the strand's latest committed record.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

La dirección del tip del resumen que proporcione debe tener un número de secuencia inferior o igual al número de secuencia del último registro confirmado de la cadena del diario. Antes de volver a intentar realizar la solicitud, asegúrese de proporcionar una dirección de tip del resumen con un número de secuencia válido.

IllegalArgumentException

Mensaje: The Strand ID of the provided block address is not valid.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

La dirección de bloque que proporcione debe tener un identificador de cadena que coincida con el identificador de cadena del diario. Antes de volver a intentar realizar la solicitud, asegúrate de proporcionar una dirección de bloque con un ID de cadena válido.

IllegalArgumentException

Mensaje: The sequence number of the provided block address is beyond the strand's latest committed record.

Operaciones de la API: `GetBlock, GetRevision`

La dirección de bloque que proporcione debe tener un número de secuencia inferior o igual al número de secuencia del último registro confirmado de la cadena. Antes de volver a intentar realizar la solicitud, asegúrese de proporcionar una dirección de bloque con un número de secuencia válido.

IllegalArgumentException

Mensaje: The Strand ID of the provided block address must match the Strand ID of the provided digest tip address.

Operaciones de la API: `GetBlock, GetRevision`

Solo puede verificar la revisión o el bloque de un documento si existe en la misma cadena del diario que el resumen que ha proporcionado.

IllegalArgumentException

Mensaje: The sequence number of the provided block address must not be greater than the sequence number of the provided digest tip address.

Operaciones de la API: `GetBlock, GetRevision`

Solo puede verificar la revisión o el bloqueo de un documento si está incluido en el resumen que proporcione. Esto significa que se registró en el diario antes que la dirección del tip del resumen.

IllegalArgumentException

Mensaje: The provided Document ID was not found in the block at the specified block address.

Operación de la API: `GetRevision`

El identificador de documento que proporcione debe estar en la dirección de bloque que proporcione. Antes de volver a intentar realizar la solicitud, asegúrese de que estos dos parámetros sean coherentes.

Exportación de datos de diarios desde Amazon QLDB

Amazon QLDB utiliza un registro transaccional inmutable, conocido como diario, para el almacenamiento de datos. El diario realiza un seguimiento de cada cambio en los datos confirmados y mantiene un historial de cambios completo y que se pueda verificar con el paso del tiempo.

Puede acceder al contenido del diario de su libro mayor para diversos fines, como el análisis, la auditoría, la retención de datos, la verificación y la exportación a otros sistemas. Los siguientes temas describen cómo exportar bloques de libro mayor a un bucket de Amazon Simple Storage Service (Amazon S3) en su Cuenta de AWS. Un trabajo de exportación de diarios escribe los datos en Amazon S3 como objetos en el formato de texto o binario de Amazon Ion, o en el formato de texto de líneas JSON.

En el formato JSON Lines, cada bloque de un objeto de datos exportado es un objeto JSON válido que está delimitado por saltos de línea. Puede usar este formato para integrar directamente las exportaciones de JSON con herramientas de análisis como Amazon Athena y AWS Glue porque estos servicios pueden analizar automáticamente el JSON delimitado por nuevas líneas. Para obtener más información sobre el formato de datos, consulte JSON Lines.

Para obtener más información acerca de Amazon S3, consulte la Guía del usuario de Amazon Simple Storage Service.

> **Note**
>
> Si especifica JSON como formato de salida de su trabajo de exportación, QLDB convierte de forma descendente los datos del diario de Ion a JSON en los objetos de datos exportados. Para obtener más información, consulte Conversión descendente a JSON.

Temas

Cómo solicitar la exportación de una diario en QLDB

Amazon QLDB proporciona una API para solicitar la exportación de sus bloques de diario para un intervalo de fechas y horas específico y para un destino de bucket de Amazon S3 específico. Un trabajo de exportación del diario puede escribir los objetos de datos en el formato de texto o binario de Amazon Ion, o en el formato de texto JSON Lines. Puede usar el AWS Management Console, un AWS SDK o el AWS Command Line Interface (AWS CLI) para crear un trabajo de exportación.

Temas

AWS Management Console

Siga estos pasos para enviar una solicitud de exportación de diarios en QLDB mediante la consola de QLDB.

Solicitud de una exportación (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Exportar.
3. Elija Crear trabajo de exportación.
4. En la página Crear trabajo de exportación, introduzca los siguientes ajustes de exportación:
   - Libro mayor: el libro mayor cuyos bloques de diario desea exportar.
   - Fecha y hora de inicio: la marca horaria de inicio incluida en la hora universal coordinada (UTC) del rango de bloques de diario que se van a exportar. Esta marca de tiempo debe ser anterior a la Fecha y hora de finalización. Si proporciona una marca de tiempo de inicio anterior a la `CreationDateTime` del libro mayor, QLDB la asigna por defecto a la `CreationDateTime` del libro mayor.
   - Fecha y hora de finalización: la marca de tiempo de finalización (UTC) exclusiva del rango de bloques de diario que se va a exportar. Estas fecha y hora no pueden estar en el futuro.

- Destino de los bloques de diario: el nombre del prefijo y el bucket de Amazon S3 en los que el trabajo de exportación escribe los objetos de datos. Utilice el siguiente formato de URI de Amazon S3.

  ```
  s3://DOC-EXAMPLE-BUCKET/prefix/
  ```

  Debe especificar un nombre de bucket de S3 y un nombre de prefijo opcional para los objetos de salida. A continuación, se muestra un ejemplo.

  ```
  s3://DOC-EXAMPLE-BUCKET/journalExport/
  ```

  Tanto el nombre como el prefijo del bucket deben cumplir con las normas y convenciones de nomenclatura de Amazon S3. Para obtener más información sobre la nomenclatura de buckets, consulte Restricciones y limitaciones de los buckets en la Guía para desarrolladores de Amazon S3. Para obtener más información sobre prefijos de nombre de clave, consulte Clave y metadatos de objetos.

  > **Note**
  >
  > No se admiten exportaciones entre regiones. El bucket de Amazon S3 especificado debe estar en el mismo lugar Región de AWS que su libro mayor.

- Cifrado S3: la configuración de cifrado que utiliza su trabajo de exportación para escribir datos en un bucket de Amazon S3. Para obtener más información sobre opciones de cifrado del servidor en Amazon S3, consulte Protección de datos mediante cifrado del servidor en la Guía para desarrolladores de Amazon S3.
  - Cifrado predeterminado del bucket: utilice la configuración de cifrado predeterminada del bucket de Amazon S3 especificado.
  - AES-256: utilice el cifrado del servidor con claves administradas por Amazon S3 (SSE-S3).
  - AWS-KMS: utilice el cifrado del lado del servidor con claves AWS KMS administradas (SSE-KMS).

    Si elige este tipo junto con la opción Elegir otra AWS KMS key, también debe especificar una clave KMS de cifrado simétrico en el siguiente formato de nombre de recurso de Amazon (ARN).

```
arn:aws:kms:aws-region:account-id:key/key-id
```

- Acceso al servicio: el rol de IAM que concede permisos de escritura de QLDB en su bucket de Amazon S3. Si corresponde, el rol de IAM también debe conceder permisos de QLDB para usar su clave de KMS.

  Para transferir un rol a QLDB al solicitar una exportación de diario, debe tener permisos para realizar la acción *iam:PassRole* en el recurso de rol de IAM.

  - Crear y utilizar un nuevo rol de servicio: deje que la consola cree un nuevo rol para usted con los permisos necesarios para el bucket de Amazon S3 especificado.
  - Utilizar un rol de servicio existente: para obtener información sobre cómo crear manualmente este rol en IAM, consulte Permisos de exportación.
- Formato de salida: el formato de salida de los datos de su diario exportados
  - Ion en texto: representación textual (predeterminada) de Amazon Ion
  - Ion binario: representación binaria de Amazon Ion
  - JSON: formato de texto JSON delimitado por saltos de línea

    Si elige JSON, QLDB convierte de forma descendente los datos del diario de Ion a JSON en los objetos de datos exportados. Para obtener más información, consulte Conversión descendente a JSON.

5. Cuando esté conforme con los ajustes, elija Crear.

   El tiempo que tarda en finalizar el trabajo de exportación varía según el tamaño de los datos. Si la solicitud se envía correctamente, la consola vuelve a la página principal de Exportación y muestra los trabajos de exportación con su estado actual.

6. Puede ver los objetos de exportación en la consola de Amazon S3.

   Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3.

   Para obtener más información sobre el formato de estos objetos de salida, consulte Salida de exportación de diarios en QLDB.

> **Note**
>
> Los trabajos de exportación caducan siete días después de completarse. Para obtener más información, consulte Caducidad del trabajo de exportación.

API DE QLDB

También puede solicitar la exportación de una revista mediante la API de Amazon QLDB con AWS un SDK o el. AWS CLI La API de QLDB ofrece las siguientes operaciones para usarlas en los programas de aplicación:

- `ExportJournalToS3`: exporta el contenido del diario dentro de un intervalo de fechas y horas de un libro mayor determinado a un bucket de Amazon S3 específico. Un trabajo de exportación escribe los objetos de datos como objetos en el formato de texto o binario de Amazon Ion, o en el formato de texto de líneas JSON.
- `DescribeJournalS3Export`: devuelve información detallada sobre un trabajo de exportación de diario. El resultado incluye su estado actual, la hora de creación y los parámetros de la solicitud de exportación original.
- `ListJournalS3Exports`: devuelve una lista de las descripciones de los trabajos de exportación de diarios de todos los libros mayores asociados a la Cuenta de AWS y a la región actuales. El resultado de cada descripción del trabajo de exportación incluye los mismos detalles que devuelve `DescribeJournalS3Export`.
- `ListJournalS3ExportsForLedger`: devuelve una lista de las descripciones de los trabajos de exportación de diarios de un libro mayor determinado. El resultado de cada descripción del trabajo de exportación incluye los mismos detalles que devuelve `DescribeJournalS3Export`.

Para obtener una descripción completa de estas operaciones de API, consulte la Referencia de la API de Amazon QLDB.

Para obtener información sobre la exportación de datos de revistas mediante el AWS CLI, consulte la Referencia de AWS CLI comandos.

Aplicación de ejemplo (Java)

Para ver ejemplos de operaciones de exportación básicas en código Java, consulte el GitHub repositorio amazon-qldb-dmv-sampleaws-samples/ -java. Para obtener instrucciones acerca de cómo

descargar e instalar esta aplicación de ejemplo, consulte Instalación de la aplicación de ejemplo Java de Amazon QLDB. Antes de solicitar una exportación, siga los pasos del 1 al 3 de Tutorial de Java para crear un libro mayor de muestra y cargarlo con datos de ejemplo.

El código del tutorial de las siguientes clases proporciona ejemplos de cómo crear una exportación, comprobar el estado de una exportación y procesar el resultado de una exportación.

Clase	Descripción
ExportJournal	Exporta los bloques de diario del libro mayor de ejemplo `vehicle-registration` para un intervalo de fechas que va desde hace 10 minutos hasta ahora. Escribe los objetos de salida en un bucket de S3 específico o crea un bucket único si no se proporciona ninguno.
DescribeJournalExport	Describe un trabajo de exportación de diarios de una `exportId` específica en el libro mayor `vehicle-registration` de ejemplo.
ListJournalExports	Devuelve una lista de las descripciones de los trabajos de exportación de diarios del libro mayor `vehicle-registration` de ejemplo.
ValidateQldbHashChain	Valida la cadena de hash del libro mayor `vehicle-registration` de ejemplo utilizando una `exportId` dada. Si no se proporciona, solicita una nueva exportación para utilizarla en la validación de la cadena de hash.

Caducidad del trabajo de exportación

Los trabajos de exportación de diarios finalizados están sujetos a un período de retención de 7 días. Una vez transcurrido este límite, se eliminan automáticamente de forma permanente. Este período de caducidad es un límite codificado y no se puede cambiar.

Después de que un trabajo de exportación completado se haya eliminado, ya no podrá utilizar la consola de QLDB ni las siguientes operaciones de API para recuperar metadatos sobre el trabajo:

- `DescribeJournalS3Export`
- `ListJournalS3Exports`
- `ListJournalS3ExportsForLedger`

Sin embargo, esta caducidad no afecta a los datos exportados en sí mismos. Todos los metadatos se conservan en los archivos de manifiesto que se escriben en las exportaciones. Esta caducidad está diseñada para proporcionar una experiencia más fluida a las operaciones de la API que enumeran los trabajos de exportación de diarios. QLDB elimina los trabajos de exportación antiguos para garantizar que solo vea las exportaciones recientes sin tener que analizar múltiples páginas de trabajos.

Salida de exportación de diarios en QLDB

Un trabajo de exportación de diarios de Amazon QLDB escribe dos archivos de manifiesto además de los objetos de datos que contienen sus bloques de diario. Todos estos archivos se guardan en el bucket de Amazon S3 que especifique en su solicitud de exportación. En las siguientes secciones se describe el formato y el contenido de cada objeto de salida.

> **Note**
>
> Si especifica JSON como formato de salida de su trabajo de exportación, QLDB convierte de forma descendente los datos del diario de Amazon Ion a JSON en los objetos de datos exportados. Para obtener más información, vaya a Conversión descendente a JSON.

Temas

- Archivos de manifiesto
- Objetos de datos
- Conversión descendente a JSON
- Biblioteca de procesador de exportación (Java)

Archivos de manifiesto

Amazon QLDB crea dos archivos de manifiesto en el bucket de S3 proporcionado para cada solicitud de exportación. El archivo de manifiesto inicial se crea en cuanto envía la solicitud de exportación. El archivo de manifiesto final se escribe una vez finalizada la exportación. Puede utilizar estos archivos para comprobar el estado de sus trabajos de exportación en Amazon S3.

El formato del contenido de los archivos de manifiesto corresponde al formato de salida solicitado para la exportación.

Manifiesto inicial

El manifiesto inicial indica que su trabajo de exportación ha comenzado. Contiene los parámetros de entrada que has transferido a la solicitud. Además del destino de Amazon S3 y los parámetros de hora de inicio y finalización de la exportación, este archivo también contiene un `exportId`. El `exportId` es un identificador único que QLDB asigna a cada trabajo de exportación.

La convención de nomenclatura de archivos es la siguiente.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.started.manifest
```

A continuación se incluye un ejemplo de un archivo de manifiesto inicial y su contenido en formato de texto Ion.

```
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"
}
```

El manifiesto inicial solo incluye el `outputFormat` si se especificó en la solicitud de exportación. Si no especifica el formato de salida, los datos exportados tomarán el formato `ION_TEXT` predeterminado.

La operación de la API DescribeJournalS3Export y el tipo de contenido de los objetos de Amazon S3 exportados también indican el formato de salida.

Manifiesto final

El manifiesto final indica que su trabajo de exportación para una cadena concreta del diario ha finalizado. El trabajo de exportación escribe un archivo de manifiesto final independiente para cada cadena.

> **Note**
>
> En Amazon QLDB, una cadena es una partición del diario del libro mayor. Actualmente, QLDB admite diarios con una sola cadena.

El manifiesto final incluye una lista ordenada de claves de objetos de datos que se escribieron durante la exportación. La convención de nomenclatura de archivos es la siguiente.

```
s3://DOC-EXAMPLE-BUCKET/prefix/exportId.strandId.completed.manifest
```

El `strandId` es un identificador único que QLDB asigna a la cadena. A continuación se incluye un ejemplo de un archivo de manifiesto final y su contenido en formato de texto Ion.

```
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"
  ]
}
```

Objetos de datos

Amazon QLDB escribe los objetos de datos del diario en el bucket Amazon S3 especificado en el formato de texto o binario de Amazon Ion, o en el formato de texto de líneas JSON.

En el formato JSON Lines, cada bloque de un objeto de datos exportado es un objeto JSON válido que está delimitado por saltos de línea. Puede usar este formato para integrar directamente las exportaciones de JSON con herramientas de análisis como Amazon Athena y AWS Glue porque estos servicios pueden analizar automáticamente el JSON delimitado por nuevas líneas. Para obtener más información sobre el formato de datos, consulte JSON Lines.

Nombres de objetos de datos

Un trabajo de exportación de diarios escribe estos objetos de datos con la siguiente convención de nomenclatura.

```
s3://DOC-EXAMPLE-BUCKET/prefix/yyyy/mm/dd/hh/strandId.startSn-endSn.ion|.json
```

- Los datos de salida de cada trabajo de exportación se dividen en partes.
- `yyyy/mm/dd/hh`: la fecha y hora en que envió la solicitud de exportación. Los objetos que se exportan en la misma hora se agrupan con el mismo prefijo de Amazon S3.
- `strandId`: el identificador único de la cadena del diario concreta que contiene el bloque del diario que se va a exportar.
- `startSn-endSn`: el rango de números de secuencia que se incluye en el objeto. Un número de secuencia especifica la ubicación de un bloque dentro de una cadena.

Por ejemplo, suponga que especifica la siguiente ruta.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/
```

El trabajo de exportación crea un objeto de datos de Amazon S3 similar al siguiente. Este ejemplo muestra el nombre de un objeto en formato Ion.

```
s3://DOC-EXAMPLE-BUCKET/journalExport/2019/04/15/22/JdxjkR9bSYB5jMHWcI464T.1-5.ion
```

Contenido del objeto de datos

Cada objeto de datos contiene objetos de bloques de diario con el siguiente formato.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
```

```
  },
  transactionId: String,
  blockTimestamp: Datetime,
  blockHash: SHA256,
  entriesHash: SHA256,
  previousBlockHash: SHA256,
  entriesHashList: [ SHA256 ],
  transactionInfo: {
    statements: [
      {
        //PartiQL statement object
      }
    ],
    documents: {
      //document-table-statement mapping object
    }
  },
  revisions: [
    {
      //document revision object
    }
  ]
}
```

Un bloque es un objeto que se asigna al diario durante una transacción. Un bloque contiene metadatos de transacciones junto con entradas que representan las revisiones del documento que se registraron en la transacción y las instrucciones PartiQL que las confirmaron.

A continuación, se muestra un ejemplo de un bloque con datos de ejemplo en formato de texto Ion. Para obtener más información sobre los campos en un objeto de bloque, consulte Contenido del diario en Amazon QLDB.

> **Note**
>
> Este ejemplo de bloque se ofrece solo con fines informativos. Los hash que se muestran no son valores hash calculados reales.

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

En el campo `revisions`, es posible que algunos objetos de revisión contengan solo un valor `hash` y ningún otro atributo. Son revisiones del sistema exclusivamente internas que no contienen datos de usuario. Un trabajo de exportación incluye estas revisiones en sus bloques respectivos porque los hashes de estas revisiones forman parte de toda la cadena de hash del diario. Se requiere la cadena de hash completa para la verificación criptográfica.

Conversión descendente a JSON

Si especifica JSON como formato de salida de su trabajo de exportación, QLDB convierte de forma descendente los datos del diario de Amazon Ion a JSON en los objetos de datos exportados. Sin

embargo, la conversión de Ion a JSON conlleva pérdidas en algunos casos en los que los datos emplean tipos de Ion enriquecidos que no existen en JSON.

Para obtener más información sobre las reglas de conversión de Ion a JSON, consulte Conversión descendente a JSON en Amazon Ion Cookbook.

Biblioteca de procesador de exportación (Java)

QLDB ofrece un marco extensible para Java que agiliza el procesamiento de las exportaciones en Amazon S3. Esta biblioteca de marcos se encarga del trabajo de leer el resultado de una exportación e iterar los bloques exportados en orden secuencial. Para usar este procesador de exportación, consulte el repositorio awslabs/ -java GitHub . amazon-qldb-export-processor

Permisos de exportación de diarios en QLDB

Antes de enviar una solicitud de exportación de diarios en Amazon QLDB, debe proporcionar a QLDB permisos de escritura en el bucket de Amazon S3 especificado. Si elige una AWS KMS key administrada por el cliente como tipo de cifrado de objetos para su bucket de Amazon S3, también debe proporcionar a QLDB permisos para utilizar la clave de cifrado simétrica especificada. Amazon S3 no admite las claves de KMS asimétricas.

Para proporcionar a su trabajo de exportación los permisos necesarios, puede hacer que QLDB asuma un rol de servicio de IAM con las políticas de permisos adecuadas. Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Note**
>
> Para transferir un rol a QLDB al solicitar una exportación de diario, debe tener permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM. Esto se suma al permiso `qldb:ExportJournalToS3` del recurso de libro mayor de QLDB.
> Para obtener información sobre cómo controlar el acceso a QLDB mediante IAM, consulte Cómo funciona Amazon QLDB con IAM. Para ver una política de ejemplo de QLDB, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

En este ejemplo, se crea un rol que permite a QLDB escribir objetos en un bucket de Amazon S3 en su nombre. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

Si va a exportar un diario de QLDB a Cuenta de AWS su cuenta por primera vez, primero debe crear un rol de IAM con las políticas adecuadas de la siguiente manera. O bien, puede usar la consola de QLDB para que cree automáticamente el rol por usted. También puede elegir un rol que haya creado anteriormente.

Temas

- Creación de una política de permisos
- Creación de un rol de IAM

Creación de una política de permisos

Complete los siguientes pasos para crear una política de permisos para un trabajo de exportación de diarios de QLDB. En este ejemplo, se muestra una política de bucket de Amazon S3 que concede permisos de QLDB para escribir objetos en el bucket especificado. Si corresponde, el ejemplo también muestra una política de claves que permite a QLDB utilizar su clave KMS de cifrado simétrica.

Para obtener más información acerca de las políticas de bucket para Amazon S3, consulte Uso de políticas de bucket y políticas de usuario en la Guía del usuario de Amazon Simple Storage Service. Para obtener más información acerca de las políticas de claves de AWS KMS , consulte Uso de las políticas de claves en AWS KMS en la Guía para desarrolladores de AWS Key Management Service .

> **Note**
>
> Tanto el bucket de Amazon S3 como la clave de KMS deben estar en el mismo lugar que el Región de AWS libro mayor de QLDB.

Utilización del editor de política de JSON para la creación de una política

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.

2. En la columna de navegación de la izquierda, elija Políticas.

   Si es la primera vez que elige Políticas, aparecerá la página Bienvenido a políticas administradas. Elija Comenzar.

3. En la parte superior de la página, seleccione Crear política.

4. Seleccione la pestaña JSON.

5. Especifique un documento de política JSON.

   - Si utiliza una clave de KMS administrada por el cliente para el cifrado de objetos de Amazon S3, utilice el siguiente documento de política de ejemplo. Para usar esta política, sustituya los *DOC-EXAMPLE-BUCKET*, *us-east-1*, *123456789012* y *1234abcd-12ab-34cd-56ef-1234567890ab* del ejemplo por su propia información.

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

   - Para otros tipos de cifrado, utilice el siguiente ejemplo de documento de política. Para usar esta política, sustituya *DOC-EXAMPLE-BUCKET* en el ejemplo por su propio nombre del bucket de Amazon S3.

     ```
     {
         "Version": "2012-10-17",
     ```

```
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permission",
            "Action": [
                "s3:PutObjectAcl",
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

6. Elija Revisar política.

> **Note**
>
> Puede alternar entre las pestañas Visual editor (Editor visual) y JSON en cualquier momento. Sin embargo, si realiza cambios o elige Review policy en la pestaña Visual editor, IAM podría reestructurar la política para optimizarla para el editor visual. Para obtener más información, consulte Reestructuración de política en la Guía del usuario de IAM.

7. En la página Review Policy (Revisar política), ingrese un Nombre y una descripción (opcional) para la política que está creando. Revise el Summary (Resumen) de la política para ver los permisos concedidos por su política. A continuación, elija Create policy (Crear política) para guardar su trabajo.

Creación de un rol de IAM

Tras crear una política de permisos para su trabajo de exportación de diarios de QLDB, puede crear un rol de IAM y adjuntarle su política.

Creación de un rol de servicio de QLDB (consola de IAM)

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola de IAM, seleccione Roles y, a continuación, elija Crear rol.

3. En Tipo de entidad de confianza, elija Servicio de AWS.
4. En Servicio o caso de uso, seleccione QLDB y, a continuación, el caso de uso QLDB.
5. Elija Siguiente.
6. Seleccione la casilla junto a la política que ha creado en los pasos anteriores.
7. (Opcional) Configure un límite de permisos. Se trata de una característica avanzada que está disponible para los roles de servicio, pero no para los roles vinculados a servicios.

   a. Abra la sección Configurar límite de permisos y, a continuación, elija Utilizar un límite de permisos para controlar los permisos que puede tener el rol como máximo.

      IAM incluye una lista de las políticas AWS gestionadas y gestionadas por los clientes de tu cuenta.

   b. Seleccione la política que desea utilizar para el límite de permisos.
8. Elija Siguiente.
9. Escriba un nombre o sufijo de nombre para el rol, que pueda ayudarle a identificar su finalidad.

   > **Important**
   >
   > Cuando asigne un nombre a un rol, tenga en cuenta lo siguiente:
   >
   > - Los nombres de los roles deben ser únicos dentro de tu perfil Cuenta de AWS y no se pueden hacer únicos por mayúsculas y minúsculas.
   >
   >   Por ejemplo, no puede crear roles denominados tanto **`PRODROLE`** como **`prodrole`**. Cuando se utiliza un nombre de rol en una política o como parte de un ARN, el nombre de rol distingue entre mayúsculas y minúsculas, sin embargo, cuando un nombre de rol les aparece a los clientes en la consola, como por ejemplo durante el proceso de inicio de sesión, el nombre de rol no distingue entre mayúsculas y minúsculas.
   > - Dado que otras entidades podrían hacer referencia al rol, no es posible editar el nombre del rol una vez creado.
10. (Opcional) En Descripción, ingrese una descripción para el rol.
11. (Opcional) Para editar los casos de uso y los permisos de la función, en las secciones Paso 1: Seleccionar entidades confiables o en Paso 2: Agregar permisos, elija Editar.

12. (Opcional) Para ayudar a identificar, organizar o buscar el rol, agregue etiquetas como pares clave-valor. Para obtener más información sobre el uso de etiquetas en IAM, consulte Etiquetado de recursos de IAM en la Guía de usuario de IAM.
13. Revise el rol y, a continuación, elija Crear rol.

El siguiente documento JSON es un ejemplo de una política de confianza que permite a QLDB asumir un rol de IAM con permisos específicos asociados.

```
{
    "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**
>
> El siguiente ejemplo de política de confianza muestra cómo se pueden utilizar las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` para evitar el problema del suplente confuso. Con esta política de confianza, QLDB solo puede asumir el rol de cualquier recurso de QLDB en la cuenta `123456789012`.
> Para obtener más información, consulte Prevención de la sustitución confusa entre servicios.

Tras crear su rol de IAM, vuelva a la consola de QLDB y actualice la página Crear trabajo de exportación para que pueda encontrar su nuevo rol.

Errores comunes en la exportación de diarios

En esta sección se describen los errores de tiempo de ejecución que genera Amazon QLDB para las solicitudes de exportación de diarios.

La siguiente es una lista de excepciones comunes devueltas por el servicio. Cada excepción incluye el mensaje de error específico, seguido de una breve descripción y sugerencias de posibles soluciones.

AccessDeniedException

> *Mensaje: Usuario: UserArn no está autorizado a realizar: iam: PassRole on recurso: roLearn*
>
> No tiene permisos para transferir un rol de IAM al servicio QLDB. QLDB requiere un rol para todas las solicitudes de exportación de diarios y debe tener permisos para transferir este rol a QLDB. El rol proporciona a QLDB permisos de escritura en el bucket de Amazon S3 especificado.
>
> Compruebe que ha definido una política de IAM que conceda permiso para realizar la operación de API `PassRole` en el recurso de rol de IAM especificado para el servicio QLDB (`qldb.amazonaws.com`). Para ver una política de ejemplo, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

IllegalArgumentException

> Mensaje: QLDB encountered an error validating S3 configuration: *errorCode errorMessage*
>
> Una posible causa de este error es que el bucket de Amazon S3 proporcionado no existe en Amazon S3. O bien, QLDB no tiene permisos suficientes para escribir objetos en el bucket de Amazon S3 especificado.
>
> Compruebe que el nombre del bucket de S3 que proporcionó en su solicitud de trabajo de exportación es correcto. Para obtener información sobre la nomenclatura de buckets, consulte Restricciones y limitaciones de los buckets en la Guía del usuario de Amazon Simple Storage Service.

Además, compruebe que ha definido una política para el bucket especificado que conceda los permisos `PutObject` y `PutObjectAcl` al servicio QLDB (`qldb.amazonaws.com`). Para obtener más información, consulte Permisos de exportación.

IllegalArgumentException

Mensaje: Unexpected response from Amazon S3 while validating the S3 configuration. Response from S3: *`errorCode errorMessage`*

El intento de escribir los datos de exportación del diario en el bucket de S3 proporcionado falló con la respuesta de error de Amazon S3 proporcionada. Para obtener más información acerca de las posibles causas, consulte Solución de problemas de Amazon S3 en la Guía de usuario de Amazon Simple Storage Service.

IllegalArgumentException

Mensaje: Amazon S3 bucket prefix must not exceed 128 characters

El prefijo proporcionado en la solicitud de exportación del diario contiene más de 128 caracteres.

IllegalArgumentException

Mensaje: Start date must not be greater than end date

`InclusiveStartTime` y `ExclusiveEndTime` deben estar en formato de fecha y hora ISO 8601 y en hora universal coordinada (UTC).

IllegalArgumentException

Mensaje: End date cannot be in the future

Tanto `InclusiveStartTime` como `ExclusiveEndTime` deben estar en formato `ISO 8601` de fecha y hora y en UTC.

IllegalArgumentException

Mensaje: La configuración de cifrado de objetos (S3EncryptionConfiguration) proporcionada no es compatible con una clave () AWS Key Management Service AWS KMS

Ha proporcionado un `KMSKeyArn` con un `ObjectEncryptionType` de `NO_ENCRYPTION` o `SSE_S3`. Solo puede proporcionar una AWS KMS key administrada por el cliente para un tipo de cifrado de objeto `SSE_KMS`. Para obtener más información sobre opciones de cifrado del servidor en Amazon S3, consulte Protección de datos mediante cifrado del servidor en la Guía para desarrolladores de Amazon S3.

LimitExceededException

Mensaje: Exceeded the limit of 2 concurrently running Journal export jobs

QLDB impone un límite predeterminado de dos trabajos de exportación de diarios simultáneos.

Transmisión de datos de diarios desde Amazon QLDB

Amazon QLDB utiliza un registro transaccional inmutable, conocido como diario, para el almacenamiento de datos. El diario realiza un seguimiento de cada cambio en los datos confirmados y mantiene un historial de cambios completo y que se pueda verificar con el paso del tiempo.

Puede crear una secuencia de QLDB que capture todas las revisiones de documentos consignadas en su diario y entregue estos datos a Amazon Kinesis Data Streams en tiempo prácticamente real. Un secuencia de QLDB es un flujo continuo de datos desde el diario de su libro mayor a un recurso de flujo de datos de Kinesis.

A continuación, puede utilizar la plataforma de transmisión de Kinesis Client Library para consumir su secuencia, procesar los registros de datos y analizar el contenido de los datos. Un secuencia de QLDB escribe sus datos en Kinesis Data Streams en tres tipos de registros: control, resumen de bloques y detalles de revisión. Para obtener más información, consulte Registros de secuencia de QLDB en Kinesis.

Temas

Casos de uso comunes

Las secuencias le permiten utilizar QLDB como una fuente de información fiable única y verificable, a la vez que integra los datos de su diario con otros servicios. Los siguientes son algunos de los casos de uso comunes admitidos por las secuencias de QLDB:

- Arquitectura basada en eventos: cree aplicaciones en un estilo arquitectónico basado en eventos con componentes disociados. Por ejemplo, un banco puede usar AWS Lambda funciones para implementar un sistema de notificación que avise a los clientes cuando el saldo de su cuenta caiga por debajo de un umbral. En un sistema de este tipo, los saldos de las cuentas se mantienen en un libro mayor de QLDB y cualquier cambio de saldo se registra en el diario. La AWS Lambda función puede activar la lógica de notificaciones al consumir un evento de actualización de saldo que se registra en el diario y se envía a una transmisión de datos de Kinesis.
- Análisis en tiempo real: cree aplicaciones de Kinesis para consumidores que ejecuten análisis en tiempo real de los datos de eventos. Con esta capacidad, puede obtener información prácticamente en tiempo real y responder rápidamente a un entorno empresarial cambiante. Por ejemplo, un sitio web de comercio electrónico puede analizar los datos de ventas de un producto y detener la publicidad de un producto con descuento tan pronto como las ventas alcancen un límite.
- Análisis histórico: aproveche la arquitectura orientada a diarios de Amazon QLDB reproduciendo datos de eventos históricos. Puede elegir iniciar una secuencia de QLDB en cualquier momento del pasado, en el que todas las revisiones realizadas desde ese momento se envíen a Kinesis Data Streams. Con esta característica, puede crear aplicaciones de Kinesis para consumidores que ejecuten trabajos de análisis con datos históricos. Por ejemplo, un sitio web de comercio electrónico puede ejecutar análisis según sea necesario para generar métricas de ventas pasadas que no se recopilaron anteriormente.
- Replicación en bases de datos personalizadas: conecte los libros mayores de QLDB con otros almacenes de datos diseñados específicamente mediante secuencias de diarios de QLDB. Por ejemplo, utilice la plataforma de datos de streaming Kinesis para integrarla con Amazon OpenSearch Service, que puede proporcionar funciones de búsqueda de texto completo para documentos QLDB. También puede crear aplicaciones Kinesis personalizadas para consumidores a fin de replicar los datos de su diario en otras bases de datos personalizadas que ofrecen diferentes vistas materializadas. Por ejemplo, replique en Amazon Aurora para datos relacionales o en Amazon Neptune para datos basados en gráficos.

Consumir la secuencia

Puede utilizar Kinesis Data Streams para consumir, procesar y analizar continuamente grandes secuencias de registros de datos. Además de Kinesis Data Streams, la plataforma de transmisión de datos Kinesis incluye Amazon Data Firehose y Amazon Managed Service para Apache Flink. Puede utilizar esta plataforma para enviar registros de datos directamente a servicios como Amazon OpenSearch Service, Amazon Redshift, Amazon S3 o Splunk. Para obtener más información,

consulte Kinesis Data Streams para consumidores en la Guía para desarrolladores de Amazon Kinesis Data Streams.

También puede usar la biblioteca de clientes de Kinesis (KCL) para crear una aplicación de consumo de secuencias que procese los registros de datos de forma personalizada. KCL simplifica la codificación porque proporciona abstracciones útiles por encima del API de bajo nivel de Kinesis Data Streams. Para obtener más información sobre KCL, consulte la sección Uso de la biblioteca de clientes Kinesis en la Guía de desarrolladores de Amazon Kinesis Data Streams.

Garantía de entrega

Las transmisiones de QLDB ofrecen at-least-onceuna garantía de entrega. Cada registro de datos generado por una secuencia de QLDB se entrega a Kinesis Data Streams al menos una vez. Los mismos registros pueden aparecer en un flujo de datos de Kinesis varias veces. Por lo tanto, debe tener una lógica de duplicación en la capa de aplicaciones de consumo si su caso de uso lo requiere.

Tampoco hay garantías de pedido. En algunas circunstancias, los bloques y las revisiones de QLDB se pueden generar en un flujo de datos de Kinesis de forma desordenada. Para obtener más información, consulte Manejo de duplicados y registros out-of-order.

Consideraciones sobre la latencia de entrega

Las secuencias de QLDB normalmente envían actualizaciones a Kinesis Data Streams casi en tiempo real. Sin embargo, los siguientes escenarios podrían crear una latencia adicional antes de que los datos de QLDB recién confirmados se emitan a un flujo de datos de Kinesis:

- Kinesis puede limitar los datos que se transmiten desde QLDB, en función del aprovisionamiento de Kinesis Data Streams. Por ejemplo, esto puede ocurrir si tiene varias secuencias de QLDB que escriben en un solo flujo de datos de Kinesis y la tasa de solicitudes de QLDB supera la capacidad del recurso de flujo de Kinesis. La limitación en Kinesis también puede producirse cuando se utiliza el aprovisionamiento bajo demanda si el rendimiento aumenta a más del doble del pico anterior en menos de 15 minutos.

  Puede medir este rendimiento superado supervisando la métrica `WriteProvisionedThroughputExceeded` de Kinesis. Para obtener más información y posibles soluciones, consulte ¿Cómo se solucionan los errores de limitación en Kinesis Data Streams?
- Con los secuencias de QLDB, puede crear un flujo indefinido con una fecha y hora de inicio pasadas y sin fecha ni hora de finalización. Por diseño, QLDB comienza a emitir los datos recién

confirmados a Kinesis Data Streams solo después de que todos los datos anteriores a la fecha y hora de inicio especificadas se hayan entregado correctamente. Si percibe una latencia adicional en este escenario, puede que tenga que esperar a que se entreguen los datos anteriores o puede iniciar el flujo a partir de una fecha y hora de inicio posteriores.

Introducción a los flujos

A continuación, se ofrece una descripción general de alto nivel de los pasos necesarios para empezar a transmitir los datos del diario a Kinesis Data Streams:

1. Creación de un recurso de Kinesis Data Streams. Para obtener instrucciones, consulte Creación y actualización de flujos de datos en la Guía para desarrolladores de Amazon Kinesis Data Streams.
2. Cree un rol de IAM que permita a QLDB asumir permisos de escritura para el flujo de datos de Kinesis. Para ver instrucciones, consulte Permisos de secuencia en QLDB.
3. Cree una secuencia de diario de QLDB. Para ver instrucciones, consulte Creación y administración de flujos en QLDB.
4. Consuma el flujo de datos de Kinesis, tal y como se describe en la sección Consumir la secuenciaanterior. Para ver ejemplos de código que muestran cómo utilizar la biblioteca de clientes de Kinesis o AWS Lambda, consulte. Desarrollo con secuencias en QLDB

Creación y administración de flujos en QLDB

Amazon QLDB proporciona operaciones de API para crear y administrar un flujo de datos de diario desde su libro mayor a Amazon Kinesis Data Streams. El secuencia de QLDB captura cada revisión del documento consignadas a su diario y la envía a un flujo de datos de Kinesis.

Puede usar el AWS Management Console, un AWS SDK o el AWS Command Line Interface (AWS CLI) para crear una transmisión de diario. Además, también puede usar una plantilla AWS CloudFormation para crear flujos. Para obtener más información, consulte el AWS::QLDB::Streamrecurso en la Guía del AWS CloudFormation usuario.

Temas

- Parámetros de flujo
- ARN de flujo
- AWS Management Console

- Estado de la secuencia
- Gestión de secuencias dañadas

Parámetros de flujo

Para crear una secuencia de diario de QLDB, debe proporcionar los siguientes parámetros de configuración:

Nombre del libro mayor

El libro mayor de QLDB cuyos datos de diario desea transmitir a Kinesis Data Streams.

Nombre de flujo

Nombre que desea asignar al flujo de diario QLDB. Los nombres definidos por el usuario pueden ayudar a identificar e indicar el propósito de un flujo.

El nombre del flujo debe ser único entre otros flujos activos de un libro mayor determinado. Los nombres de los flujos tienen las mismas restricciones de denominación que los nombres de los libros mayores, tal como se definen en Cuotas y límites de Amazon QLDB.

Además del nombre del flujo, QLDB asigna un ID de flujo a cada secuencia de QLDB que cree. El ID de flujo es único entre todos los flujos de un registro determinado, independientemente de su estado.

Fecha y hora de inicio

La fecha y la hora a partir de las cuales se iniciará el flujo de datos del diario. Este valor puede ser cualquier fecha y hora del pasado, pero no puede estar en el futuro.

Fecha y hora de finalización

(Opcional) La fecha y hora exclusivas que especifican cuándo termina el flujo.

Si crea un flujo indefinido sin hora de finalización, debe cancelarlo manualmente para finalizarlo. También puede cancelar un flujo finito y activo que aún no haya alcanzado la fecha y hora de finalización especificadas.

Flujo de datos de Kinesis de destino

El recurso de destino de Kinesis Data Streams en el que el flujo escribe los registros de datos. Para obtener información acerca de la creación de un flujo de datos de Kinesis, consulte Creación

y actualización de secuencias de datos en la Guía para desarrolladores de Amazon Kinesis Data Streams.

> **Important**
>
> - No se admite el flujo entre cuentas ni entre regiones. El flujo de datos de Kinesis especificado debe estar en las mismas Región de AWS y cuenta que su libro mayor.
> - La agregación de registros en Kinesis Data Streams está habilitada de forma predeterminada. Esta opción permite a QLDB publicar varios registros de datos en un único registro de Kinesis Data Streams, lo que aumenta el número de registros enviados por llamada a la API.
>
>   La agregación de registros tiene importantes implicaciones para el procesamiento de registros y requiere la desagrupación en su consumidor de flujos. Para obtener más información, consulte Conceptos clave de KPL y Desagrupación del consumidor en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Rol de IAM

El rol de IAM que permite a QLDB asumir permisos de escritura en su flujo de datos de Kinesis. Puede usar la consola de QLDB para crear automáticamente este rol o puede crearlo manualmente en IAM. Para aprender a crearlo manualmente, consulte Permisos de secuencia.

Para transferir un rol a QLDB al solicitar una secuencia, debe tener permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM.

ARN de flujo

Cada flujo de diario de QLDB es un subrecurso de un libro mayor y se identifica de forma inequívoca mediante un nombre de recurso de Amazon (ARN). El siguiente es un ejemplo de ARN de una secuencia de QLDB con un ID de flujo de `IiPT4brpZCqCq3f4MTHbYy` para un libro mayor denominado `exampleLedger`.

```
arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/IiPT4brpZCqCq3f4MTHbYy
```

En la siguiente sección se describe cómo crear y cancelar una secuencia de QLDB mediante AWS Management Console.

AWS Management Console

Siga estos pasos para crear o cancelar una secuencia de QLDB mediante la consola de QLDB.

Creación de una etiqueta (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. Elija Flujos en el panel de navegación.
3. Seleccione Crear la secuencia de QLDB.
4. En la página Crear la secuencia de QLDB, introduzca los siguientes ajustes:

   - Nombre de la secuencia: nombre que desea asignar a la secuencia de diario QLDB.
   - Libro mayor: el libro mayor cuyos datos del diario desea transmitir.
   - Fecha y hora de inicio: marca de tiempo inclusiva en la Hora universal coordinada (UTC) a partir de la cual se iniciará el flujo de datos del diario. Esta marca de hora tiene por defecto la hora y fecha actuales. No puede ser en el futuro y debe ser anterior a la fecha y hora de finalización.
   - Fecha y hora de finalización: (opcional) la marca de tiempo exclusiva (UTC) que especifica cuándo finaliza el flujo. Si no define este parámetro, el flujo se ejecutará indefinidamente hasta que lo cancele.
   - Secuencia de destino: el recurso de destino de Kinesis Data Streams en el que el flujo escribe los registros de datos. Use el siguiente formato ARN.

     ```
     arn:aws:kinesis:aws-region:account-id:stream/kinesis-stream-name
     ```

     A continuación, se muestra un ejemplo.

     ```
     arn:aws:kinesis:us-east-1:123456789012:stream/stream-for-qldb
     ```

     No se admite el flujo entre cuentas ni entre regiones. El flujo de datos de Kinesis especificado debe estar en la misma cuenta Región de AWS y cuenta que su libro mayor.
   - Habilitar la agregación de registros en Kinesis Data Streams: (habilitada de forma predeterminada) permite a QLDB publicar varios registros de datos en un único registro de Kinesis Data Streams, lo que aumenta el número de registros enviados por llamada a la API.

- Acceso al servicio: el rol de IAM que otorga permisos de escritura de QLDB a su flujo de datos de Kinesis.

  Para transferir un rol a QLDB al solicitar una secuencia, debe tener permisos para realizar la acción *`iam:PassRole`* en el recurso de rol de IAM.

  - Cree y utilice un nuevo rol de servicio: deje que la consola cree un nuevo rol para usted con los permisos necesarios para el flujo de datos de Kinesis especificado.
  - Utilizar un rol de servicio existente: para obtener información sobre cómo crear manualmente este rol en IAM, consulte Permisos de secuencia.

- Etiquetas: (opcional) agregue metadatos al flujo asociando etiquetas como pares de clave-valor. Puede añadir etiquetas a los flujos para facilitar su organización e identificación. Para obtener más información, consulte Etiquetado de recursos de Amazon QLDB.

  Elija Agregar etiqueta y, a continuación, introduzca cualquier par clave-valor según corresponda.

5. Cuando esté conforme con los ajustes, elija Crear secuencia de QLDB.

   Si la solicitud se envía correctamente, la consola vuelve a la página principal de Secuencias y muestra las secuencias QLDB con su estado actual.

6. Una vez que la secuencia esté activa, utilice Kinesis para procesar los datos del flujo con una aplicación para consumidores.

   Abra la consola Kinesis Data Streams en https://console.aws.amazon.com/kinesis.

Para obtener información sobre el formato de los registros de datos de secuencia, consulte Registros de secuencia de QLDB en Kinesis.

Para obtener información sobre cómo administrar las secuencias que provocan un error, consulte Gestión de secuencias dañadas.

Cómo cancelar una etiqueta (consola)

No puede reiniciar una secuencia de QLDB después de cancelarlo. Para reanudar la entrega de sus datos a Kinesis Data Streams, puede crear una nueva secuencia de QLDB.

1. Abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. Elija Flujos en el panel de navegación.

3. En la lista de secuencias de QLDB, seleccione la secuencia activa que desea cancelar.
4. Seleccione Cancelar paso. Confirme ingresando **`cancel stream`** en el cuadro provisto.

Para obtener información sobre el uso de la API de QLDB con AWS un SDK o AWS CLI para crear y administrar transmisiones de diarios, consulte. Desarrollo con secuencias en QLDB

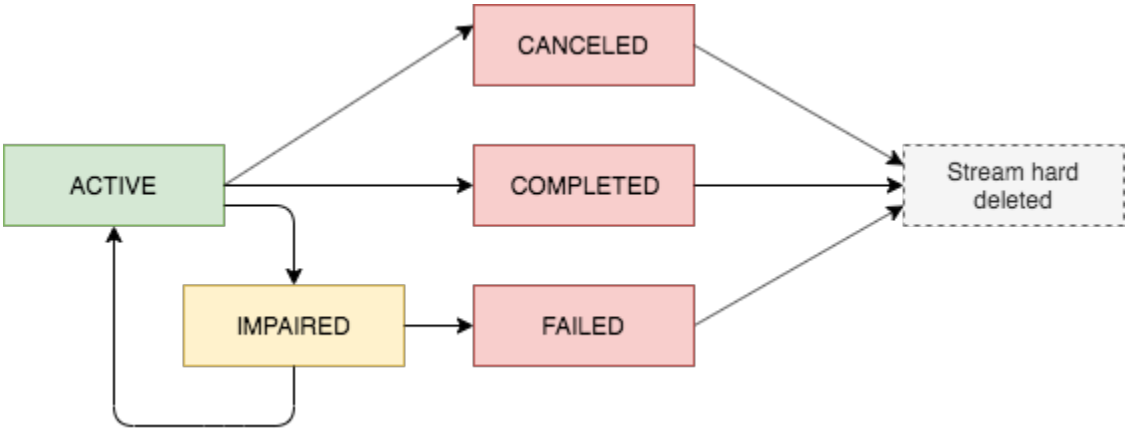
Estado de la secuencia

El estado de una secuencia de QLDB puede ser uno de los siguientes:

- `ACTIVE`: está transmitiendo datos actualmente o esperando para transmitirlos (si se trata de una secuencia indefinida sin hora de finalización).
- `COMPLETED`: ha finalizado correctamente la secuencia de todos los bloques del diario dentro del intervalo de tiempo especificado. Se trata de un estado terminal.
- `CANCELED`: ha sido finalizada por una solicitud de un usuario antes de la hora de finalización especificada y ya no está transmitiendo datos de forma activa. Se trata de un estado terminal.
- `IMPAIRED`: no puede escribir registros en Kinesis debido a un error que requiere su acción. Se trata de un estado recuperable y no terminal.

  Si resuelve el error en el plazo de una hora, la secuencia pasará automáticamente al estado `ACTIVE`. Si el error sigue sin resolverse después de una hora, la secuencia pasa automáticamente al estado `FAILED`.
- `FAILED`: no puede escribir registros en Kinesis debido a un error y se encuentra en un estado terminal irrecuperable.

El siguiente diagrama ilustra cómo un recurso de secuencia de QLDB puede realizar la transición entre estados.



Vencimiento de las secuencias terminales

Los recursos de secuencia que se encuentran en un estado terminal (`CANCELED`, `COMPLETED` y `FAILED`) están sujetos a un período de retención de 7 días. Una vez transcurrido este límite, se eliminan automáticamente de forma permanente.

Después de eliminar una secuencia terminal, ya no podrá utilizar la consola de QLDB o la API de QLDB para describir o enumerar el recurso de secuencia.

Gestión de secuencias dañadas

Si la secuencia encuentra un error, pasará primero al estado `IMPAIRED`. QLDB continúa reintentando las secuencias `IMPAIRED` durante un máximo de una hora.

Si resuelve el error en el plazo de una hora, la secuencia pasará automáticamente al estado `ACTIVE`. Si el error sigue sin resolverse después de una hora, la secuencia pasa automáticamente al estado `FAILED`.

Una secuencia defectuosa o fallida puede tener una de las siguientes causas de error:

- `KINESIS_STREAM_NOT_FOUND`: el recurso de Kinesis Data Streams de destino no existe. Compruebe que el flujo de datos de Kinesis que proporcionó en su solicitud de secuencia de QLDB sea correcto. A continuación, vaya a Kinesis y cree el flujo de datos que especificó.
- `IAM_PERMISSION_REVOKED`: QLDB no tiene permisos suficientes para escribir registros de datos en el flujo de datos de Kinesis especificado. Compruebe que ha definido una política para el flujo de datos de Kinesis especificado que conceda al servicio QLDB permisos (`qldb.amazonaws.com`) para las siguientes acciones:
  - `kinesis:PutRecord`

- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

Monitorización de secuencias dañadas

Si una secuencia se interrumpe, la consola de QLDB muestra un banner con detalles sobre la secuencia y el error que ha encontrado. También puede utilizar la operación de la API `DescribeJournalKinesisStream` para obtener el estado de una secuencia y la causa del error subyacente.

Además, puedes usar Amazon CloudWatch para crear una alarma que monitorice la `IsImpaired` métrica de una transmisión. Para obtener información sobre la supervisión de las métricas CloudWatch de QLDB con, consulte. Dimensiones y métricas de Amazon QLDB

Desarrollo con secuencias en QLDB

En esta sección se resumen las operaciones de la API que puede utilizar con un AWS SDK o AWS CLI para crear y gestionar transmisiones de diarios en Amazon QLDB. También describe los ejemplos de aplicaciones que demuestran estas operaciones y utilizan la biblioteca de clientes de Kinesis (KCL) o AWS Lambda para implementar un consumidor de secuencias.

Puede utilizar la KCL para crear aplicaciones de consumo para Amazon Kinesis Data Streams. KCL simplifica la codificación porque proporciona abstracciones útiles por encima del API de bajo nivel de Kinesis Data Streams. Para obtener más información sobre KCL, consulte la sección Uso de la biblioteca de clientes Kinesis en la Guía de desarrolladores de Amazon Kinesis Data Streams.

Contenido

API de secuencia de diarios de QLDB

La API de QLDB ofrece las siguientes operaciones de secuencia de diarios para usarlas en los programas de aplicación:

- `StreamJournalToKinesis`: crea una secuencia de diario para un libro mayor de QLDB determinado. La secuencia captura cada revisión del documento consignada al diario de su libro mayor y entrega los datos a un recurso de Kinesis Data Streams especificado.
  - La agregación de registros en Kinesis Data Streams está habilitada de forma predeterminada. Esta opción permite a QLDB publicar varios registros de datos en un único registro de Kinesis Data Streams, lo que aumenta el número de registros enviados por llamada a la API.

    La agregación de registros tiene importantes implicaciones para el procesamiento de registros y requiere la desagrupación en su consumidor de flujos. Para obtener más información, consulte Conceptos clave de KPL y Desagrupación del consumidor en la Guía para desarrolladores de Amazon Kinesis Data Streams.
- `DescribeJournalKinesisStream`: devuelve información detallada sobre una secuencia de diario determinada de QLDB. El resultado incluye el ARN, el nombre de la secuencia, el estado actual, la hora de creación y los parámetros de la solicitud de creación de la secuencia original.
- `ListJournalKinesisStreamsForLedger`: devuelve una lista de los descriptores de las secuencias de diarios QLDB de un libro mayor determinado. El resultado de cada descriptor de secuencia incluye los mismos detalles que devuelve `DescribeJournalKinesisStream`.
- `CancelJournalKinesisStream`: finaliza una secuencia de diario de QLDB determinado. Para poder cancelar una secuencia, su estado actual debe ser `ACTIVE`.

  No puede reiniciar una secuencia después de cancelarla. Para reanudar la entrega de sus datos a Kinesis Data Streams, puede crear una nueva secuencia de QLDB.

Para obtener una descripción completa de estas operaciones de API, consulte la Referencia de la API de Amazon QLDB.

Para obtener información sobre cómo crear y administrar transmisiones de diarios mediante el AWS CLI, consulte la AWS CLI Referencia de comandos.

Aplicaciones de muestra

QLDB ofrece ejemplos de aplicaciones que demuestran diversas operaciones mediante secuencias de diarios. Estas aplicaciones son de código abierto en el GitHub sitio de AWS muestras.

Temas

- Operaciones básicas (Java)
- Integración con el OpenSearch servicio (Python)
- Integración con Amazon SNS y Amazon SQS (Python)

Operaciones básicas (Java)

Para ver un ejemplo de código Java que muestre las operaciones básicas de los flujos de diarios de QLDB, consulte el repositorio aws-samples/ GitHub -java. amazon-qldb-dmv-sample Para obtener instrucciones acerca de cómo descargar e instalar esta aplicación de ejemplo, consulte Instalación de la aplicación de ejemplo Java de Amazon QLDB.

> **Note**
>
> Tras instalar la aplicación, no continúe con el paso 1 del tutorial de Java para crear un libro mayor. Este ejemplo de aplicación para streaming crea el libro mayor `vehicle-registration` automáticamente.

Esta aplicación de ejemplo empaqueta el código fuente completo de Tutorial de Java y sus dependencias, incluidos los siguientes módulos:

- AWS SDK for Java: para crear y eliminar los recursos de QLDB y Kinesis Data Streams, incluidos los libros mayores, las secuencias de diarios de QLDB y los flujos de datos de Kinesis.
- Controlador Amazon QLDB para Java: para ejecutar transacciones de datos en un libro mayor mediante instrucciones PartiQL, incluida la creación de tablas y la inserción de documentos.
- Biblioteca de clientes de Kinesis: para consumir y procesar datos de un flujo de datos de Kinesis.

Ejecutar el código

La StreamJournalclase contiene un código tutorial que muestra las siguientes operaciones:

1. Cree un libro mayor llamado `vehicle-registration`, cree tablas y cárguelas con datos de ejemplo.

   > **Note**
   >
   > Antes de ejecutar este código, asegúrese de que aún no tiene un libro activo denominado `vehicle-registration`.

2. Cree un flujo de datos de Kinesis, un rol de IAM que permita a QLDB asumir permisos de escritura para el flujo de datos de Kinesis y una secuencia de diario de QLDB.
3. Utilice la KCL para iniciar un lector de secuencias que procese el flujo de datos de Kinesis y registre cada registro de datos de QLDB.
4. Utilice los datos de la secuencia para validar la cadena de hash del libro mayor de ejemplo `vehicle-registration`.
5. Limpie todos los recursos deteniendo el lector de secuencias, cancelando la secuencia de diarios de QLDB, eliminando el libro mayor y eliminando el flujo de datos de Kinesis.

Para ejecutar el código del tutorial `StreamJournal`, introduzca el siguiente comando de Gradle desde el directorio raíz del proyecto.

```
./gradlew run -Dtutorial=streams.StreamJournal
```

Integración con el OpenSearch servicio (Python)

Para ver un ejemplo de aplicación en Python que muestre cómo integrar una transmisión de QLDB con OpenSearch Amazon Service, consulte GitHub el repositorio aws-samples/ -. amazon-qldb-streaming-amazon opensearch-service-sample-python Esta aplicación utiliza una AWS Lambda función para implementar un consumidor de Kinesis Data Streams.

Para clonar el repositorio, introduzca el siguiente comando `git`.

```
git clone https://github.com/aws-samples/amazon-qldb-streaming-amazon-opensearch-
service-sample-python.git
```

Para ejecutar la aplicación de ejemplo, consulte las instrucciones en GitHub el archivo README.

Integración con Amazon SNS y Amazon SQS (Python)

Para ver un ejemplo de aplicación de Python que muestre cómo integrar una transmisión de QLDB con Amazon Simple Notification Service (Amazon SNS), consulte el repositorio aws-samples/ -. GitHub amazon-qldb-streams-dmv sample-lambda-python

Esta aplicación utiliza una AWS Lambda función para implementar un consumidor de Kinesis Data Streams. Envía mensajes a un tema de Amazon SNS que tiene una cola de Amazon Simple Queue Service (Amazon SQS) suscrita.

Para clonar el repositorio, introduzca el siguiente comando `git`.

```
git clone https://github.com/aws-samples/amazon-qldb-streams-dmv-sample-lambda-
python.git
```

Para ejecutar la aplicación de ejemplo, consulte las instrucciones en GitHub el archivo README.

Registros de secuencia de QLDB en Kinesis

Una secuencia de Amazon QLDB escribe tres tipos de registros de datos en un recurso de Amazon Kinesis Data Streams determinado: control, resumen de bloques y detalles de revisión. Los tres tipos de registros se escriben en la representación binaria del formato Amazon Ion.

Los registros de control indican el inicio y la finalización de sus secuencias de QLDB. Cada vez que se confirma una revisión en su diario, una secuencia de QLDB escribe todos los datos de bloques del diario asociados en los registros de resumen de bloques y detalles de la revisión.

Los tres tipos de registros son polimórficos. Todos constan de un registro común de nivel superior que contiene el ARN de la secuencia de QLDB, el tipo de registro y la carga útil del registro. Este registro de nivel superior tiene el formato siguiente.

```
{
  qldbStreamArn: string,
  recordType: string,
  payload: {
    //control | block summary | revision details record
  }
}
```

El campo `recordType` puede tener uno de estos tres valores:

- `CONTROL`
- `BLOCK_SUMMARY`
- `REVISION_DETAILS`

En las siguientes secciones se describen el formato y el contenido de cada registro de carga útil individual.

> **Note**
>
> QLDB escribe todos los registros de secuencia en Kinesis Data Streams en la representación binaria de Amazon Ion. Los siguientes ejemplos se incluyen en la representación textual de Ion para ilustrar el contenido del registro en un formato legible.

Temas

Registros de control

Un secuencia de QLDB escribe registros de control para indicar sus eventos de inicio y finalización. Los siguientes son ejemplos de registros de control con datos de muestra para cada `controlRecordType`:

- `CREATED`: el primer registro que una secuencia de QLDB escribe en Kinesis para indicar que la secuencia recién creada está activa.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"CREATED"
  }
```

```
}
```

- `COMPLETED`: el último registro que una secuencia de QLDB escribe en Kinesis para indicar que la secuencia ha alcanzado la fecha y hora de finalización especificadas. Este registro no se escribe si se cancela la secuencia.

```
{
  qldbStreamArn:"arn:aws:qldb:us-east-1:123456789012:stream/exampleLedger/
IiPT4brpZCqCq3f4MTHbYy",
  recordType:"CONTROL",
  payload:{
    controlRecordType:"COMPLETED"
  }
}
```

Registros de resumen de bloque

Un registro de resumen de bloque representa un bloque de diario en el que se consignan las revisiones del documento. Un bloque es un objeto que se asigna al diario QLDB durante una transacción.

La carga útil de un registro de resumen de bloque contiene la dirección del bloque, la marca de tiempo y otros metadatos de la transacción que confirmó el bloque. También incluye los atributos de resumen de las revisiones del bloque y las instrucciones PartiQL que las confirmaron. El siguiente es un ejemplo de un resumen de bloque con datos de muestra.

> **Note**
>
> Este ejemplo de resumen de bloque se ofrece solo con fines informativos. Los hash que se muestran no son valores hash calculados reales.

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

En el campo `revisionSummaries`, es posible que algunas revisiones no tengan un `documentId`. Son revisiones del sistema exclusivamente internas que no contienen datos de usuario. Un secuencia de QLDB incluye estas revisiones en sus respectivos registros de resumen de bloques porque los hashes de estas revisiones forman parte de la cadena de hash completa del diario. Se requiere la cadena de hash completa para la verificación criptográfica.

Solo las revisiones que tienen un identificador de documento se publican en registros de detalles de revisión independientes, como se describe en la siguiente sección.

Registros de detalles de revisión

Un registro de detalles de la revisión representa una revisión de un documento que está archivada en su diario. La carga útil contiene todos los atributos de la vista confirmada de la revisión, junto con el nombre de la tabla y el ID de la tabla asociados. El siguiente es un ejemplo de un registro de revisión con datos de muestra.

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

Manejo de duplicados y registros out-of-order

Las transmisiones de QLDB pueden publicar registros out-of-order y duplicados en Kinesis Data Streams. Por lo tanto, es posible que una aplicación consumidora necesite implementar su propia lógica para identificar y administrar estos escenarios. Los registros de resumen de bloques y detalles de la revisión incluyen campos que puede utilizar para este fin. En combinación con las características de los servicios posteriores, estos campos pueden indicar tanto una identidad única como un orden estricto para los registros.

Por ejemplo, considere un flujo que integre la QLDB con OpenSearch un índice para proporcionar capacidades de búsqueda de texto completo en los documentos. En este caso de uso, debe evitar indexar las revisiones obsoletas (out-of-order) de un documento. Para exigir el orden y la deduplicación, puede utilizar los campos de ID del documento y de versión externa OpenSearch, junto con los campos de ID de documento y versión, en un registro de detalles de la revisión.

Para ver un ejemplo de la lógica de deduplicación en una aplicación de muestra que integra QLDB con OpenSearch Amazon Service, GitHub consulte el repositorio aws-samples/ -. amazon-qldb-streaming-amazon opensearch-service-sample-python

Permisos de secuencia en QLDB

Antes de crear una secuencia de Amazon QLDB, debe proporcionar a QLDB permisos de escritura para el recurso de Amazon Kinesis Data Streams especificado. Si utiliza una AWS KMS key administrada por el cliente para el cifrado del servidor de su secuencia de Kinesis, también debe proporcionar a QLDB permisos para utilizar la clave de cifrado simétrica especificada. Kinesis Data Streams no admite claves de KMS asimétricas.

Para proporcionar a su secuencia de QLDB los permisos necesarios, puede hacer que QLDB asuma un rol de servicio de IAM con las políticas de permisos adecuadas. Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Note**
>
> Para transferir un rol a QLDB al solicitar una secuencia, debe tener permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM. Esto se suma al permiso `qldb:StreamJournalToKinesis` del subrecurso de secuencia de QLDB.
> Para obtener información sobre cómo controlar el acceso a QLDB mediante IAM, consulte Cómo funciona Amazon QLDB con IAM. Para ver una política de ejemplo de QLDB, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

En este ejemplo, crea un rol que permite a QLDB escribir registros de datos en un flujo de datos de Kinesis en su nombre. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

Si está transmitiendo un diario de QLDB en Cuenta de AWS su sitio por primera vez, primero debe crear un rol de IAM con las políticas adecuadas de la siguiente manera. O bien, puede usar la consola de QLDB para que cree automáticamente el rol por usted. También puede elegir un rol que haya creado anteriormente.

Temas

- Creación de una política de permisos
- Creación de un rol de IAM

Creación de una política de permisos

Complete los siguientes pasos para crear políticas de permisos para la secuencia de QLDB. Este ejemplo muestra una política de Kinesis Data Streams que concede permisos de QLDB para escribir registros de datos en el flujo de datos de Kinesis especificado. Si corresponde, el ejemplo también muestra una política de claves que permite a QLDB utilizar su clave KMS de cifrado simétrica.

Para obtener más información sobre Kinesis Data Streams, consulte Control del acceso a los recursos de Amazon Kinesis Data Streams por medio de IAM y Permisos de para utilizar claves de KMS generadas por el usuario en la Guía para desarrolladores de Amazon Kinesis Data Streams. Para obtener más información sobre las políticas AWS KMS clave, consulte Uso de políticas clave AWS KMS en la AWS Key Management Service Guía para desarrolladores.

> **Note**
>
> El flujo de datos de Kinesis y la clave de KMS deben estar en la misma cuenta Región de AWS y en la misma cuenta que su libro mayor de QLDB.

Utilización del editor de política de JSON para la creación de una política

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.
2. En la columna de navegación de la izquierda, elija Políticas.

   Si es la primera vez que elige Políticas, aparecerá la página Bienvenido a políticas administradas. Elija Comenzar.
3. En la parte superior de la página, seleccione Crear política.
4. Seleccione la pestaña JSON.
5. Especifique un documento de política JSON.
   - Si utiliza una clave KMS administrada por el cliente para el cifrado del servidor de su secuencia de Kinesis, utilice el siguiente documento de política de ejemplo. Para usar esta política, sustituya *us-east-1*, 123456789012 y

1234abcd-12ab-34cd-56ef-1234567890ab en el *kinesis-stream-name*ejemplo por su *propia* información.

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

- También puede utilizar el siguiente documento normativo de ejemplo. Para usar esta política, sustituya *us-east-1*, 123456789012 *kinesis-stream-name*y, en el ejemplo, por su propia información.

```
{
    "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. Elija Revisar política.

> **Note**
>
> Puede alternar entre las pestañas Visual editor (Editor visual) y JSON en cualquier momento. Sin embargo, si realiza cambios o elige Review policy en la pestaña Visual editor, IAM podría reestructurar la política para optimizarla para el editor visual. Para obtener más información, consulte Reestructuración de política en la Guía del usuario de IAM.

7. En la página Review Policy (Revisar política), ingrese un Nombre y una descripción (opcional) para la política que está creando. Revise el Summary (Resumen) de la política para ver los permisos concedidos por su política. A continuación, elija Create policy (Crear política) para guardar su trabajo.

Creación de un rol de IAM

Tras crear una política de permisos para su secuencia de QLDB, puede crear un rol de IAM y asociarle su política.

Creación de un rol de servicio de QLDB (consola de IAM)

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola de IAM, seleccione Roles y, a continuación, elija Crear rol.
3. En Tipo de entidad de confianza, elija Servicio de AWS.
4. En Servicio o caso de uso, seleccione QLDB y, a continuación, el caso de uso QLDB.
5. Elija Siguiente.
6. Seleccione la casilla junto a la política que ha creado en los pasos anteriores.
7. (Opcional) Configure un límite de permisos. Se trata de una característica avanzada que está disponible para los roles de servicio, pero no para los roles vinculados a servicios.

    a. Abra la sección Configurar límite de permisos y, a continuación, elija Utilizar un límite de permisos para controlar los permisos que puede tener el rol como máximo.

    IAM incluye una lista de las políticas AWS gestionadas y gestionadas por los clientes de tu cuenta.

b. Seleccione la política que desea utilizar para el límite de permisos.

8. Elija Siguiente.
9. Escriba un nombre o sufijo de nombre para el rol, que pueda ayudarle a identificar su finalidad.

> **Important**
>
> Cuando asigne un nombre a un rol, tenga en cuenta lo siguiente:
>
> - Los nombres de los roles deben ser únicos dentro de tu perfil Cuenta de AWS y no se pueden hacer únicos por mayúsculas y minúsculas.
>
> Por ejemplo, no puede crear roles denominados tanto **`PRODROLE`** como **`prodrole`**. Cuando se utiliza un nombre de rol en una política o como parte de un ARN, el nombre de rol distingue entre mayúsculas y minúsculas, sin embargo, cuando un nombre de rol les aparece a los clientes en la consola, como por ejemplo durante el proceso de inicio de sesión, el nombre de rol no distingue entre mayúsculas y minúsculas.
> - Dado que otras entidades podrían hacer referencia al rol, no es posible editar el nombre del rol una vez creado.

10. (Opcional) En Descripción, ingrese una descripción para el rol.
11. (Opcional) Para editar los casos de uso y los permisos de la función, en las secciones Paso 1: Seleccionar entidades confiables o en Paso 2: Agregar permisos, elija Editar.
12. (Opcional) Para ayudar a identificar, organizar o buscar el rol, agregue etiquetas como pares clave-valor. Para obtener más información sobre el uso de etiquetas en IAM, consulte Etiquetado de recursos de IAM en la Guía de usuario de IAM.
13. Revise el rol y, a continuación, elija Crear rol.

El siguiente documento JSON es un ejemplo de una política de confianza que permite a QLDB asumir un rol de IAM con permisos específicos asociados.

```
{
    "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**
>
> El siguiente ejemplo de política de confianza muestra cómo se pueden utilizar las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` para evitar el problema del suplente confuso. Con esta política de confianza, QLDB puede asumir el rol de cualquier secuencia de QLDB en la cuenta `123456789012` del libro mayor `myExampleLedger` únicamente.
>
> Para obtener más información, consulte Prevención de la sustitución confusa entre servicios.

Tras crear su rol de IAM, vuelva a la consola de QLDB y actualice la página Crear secuencia de QLDB para que pueda encontrar su nuevo rol.

Errores comunes en las secuencias de diarios en QLDB

En esta sección se describen los errores de tiempo de ejecución que genera Amazon QLDB para las solicitudes de secuencia de diarios.

La siguiente es una lista de excepciones comunes devueltas por el servicio. Cada excepción incluye el mensaje de error específico, seguido de una breve descripción y sugerencias de posibles soluciones.

AccessDeniedException

Mensaje: Usuario: UserArn no está autorizado a realizar: iam: PassRole on recurso: roLearn

No tiene permisos para transferir un rol de IAM al servicio QLDB. QLDB requiere un rol para todas las solicitudes de secuencia del diario y debe tener permisos para transferir este rol a QLDB. El rol proporciona a QLDB permisos de escritura en el recurso Amazon Kinesis Data Streams especificado.

Compruebe que ha definido una política de IAM que conceda permiso para realizar la operación de API `PassRole` en el recurso de rol de IAM especificado para el servicio QLDB (`qldb.amazonaws.com`). Para ver una política de ejemplo, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

IllegalArgumentException

Mensaje: QLDB encountered an error validating Kinesis Data Streams: Response from Kinesis: *errorCode errorMessage*

Una posible causa de este error es que el recurso de Kinesis Data Streams proporcionado no existe. O bien, QLDB no tiene permisos suficientes para escribir registros de datos en el flujo de datos de Kinesis especificado.

Compruebe que el flujo de datos de Kinesis que proporciona en su solicitud de transmisión es correcto. Para obtener más información, consulte Creación y actualización de flujos de datos en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Además, compruebe que ha definido una política para el flujo de datos de Kinesis especificado que conceda al servicio QLDB (`qldb.amazonaws.com`) permisos para las siguientes acciones. Para obtener más información, consulte Permisos de secuencia.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Mensaje: Unexpected response from Kinesis Data Streams while validating the Kinesis configuration. Response from Kinesis: *errorCode errorMessage*

El intento de escribir los registros de datos en el flujo de datos de Kinesis proporcionado falló con la respuesta de error de Kinesis proporcionada. Para obtener más información sobre las posibles causas, consulte Solución de problemas de los productores de Amazon Kinesis Data Streams en la Guía para desarrolladores de Amazon Kinesis Data Streams.

IllegalArgumentException

Mensaje: Start date must not be greater than end date.

`InclusiveStartTime` y `ExclusiveEndTime` deben estar en formato de fecha y hora ISO 8601 y en hora universal coordinada (UTC).

IllegalArgumentException

Mensaje: Start date cannot be in the future.

Tanto `InclusiveStartTime` como `ExclusiveEndTime` deben estar en formato `ISO 8601` de fecha y hora y en UTC.

LimitExceededException

Mensaje: Exceeded the limit of 5 concurrently running Journal streams to Kinesis Data Streams

QLDB impone un límite predeterminado de cinco secuencias de diarios simultáneas.

Gestión de libros mayores en Amazon QLDB

En este capítulo se describe cómo usar la API de QLDB, AWS Command Line Interface (AWS CLI) y AWS CloudFormation para llevar a cabo operaciones de gestión de libros mayores en Amazon QLDB.

También puede llevar a cabo las mismas tareas utilizando la AWS Management Console. Para obtener más información, consulte Acceso a Amazon QLDB mediante la consola.

Temas

- Operaciones básicas de libros mayores de Amazon QLDB
- Crear recursos de Amazon QLDB con AWS CloudFormation
- Etiquetado de recursos de Amazon QLDB

Operaciones básicas de libros mayores de Amazon QLDB

Puede usar la API de QLDB o AWS Command Line Interface (AWS CLI) para crear, actualizar y eliminar libros mayores en Amazon QLDB. También puede enumerar todos los libros mayores de su cuenta u obtener información sobre uno concreto.

Los siguientes temas incluyen ejemplos de código corto que muestran los pasos habituales de las operaciones de libro mayor en AWS SDK for Java y la AWS CLI.

Temas

- Creación de un libro mayor
- Descripción de un libro mayor
- Actualización de un libro mayor
- Actualizar el modo de permisos de un libro mayor
- Eliminación de un libro mayor
- Enumerar libros mayores

Para ver ejemplos de código que demuestren estas operaciones en una aplicación de muestra completa, consulte los siguientes tutoriales de Introducción al controlador y repositorios de GitHub:

- Java: Tutorial | Repositorio GitHub

- Node.js: Tutorial | Repositorio GitHub
- Python: Tutorial | Repositorio GitHub

Creación de un libro mayor

Use la operación `CreateLedger` para crear un libro mayor en su Cuenta de AWS. Debe proporcionar la siguiente información:

- Nombre de libro mayor: el nombre del libro mayor que desea crear en su cuenta. El nombre debe ser único entre todos los libros mayores de la Región de AWS actual.

  Las restricciones de nomenclatura de libros mayores se definen en Cuotas y límites de Amazon QLDB.
- Modo de permisos: el modo de permisos que se va a asignar al libro mayor. Elija una de las siguientes opciones:
  - Permitir todo: un modo de permisos heredado que habilita el control de acceso con granularidad de la API para los libros mayores.

    Este modo permite a los usuarios que tengan el permiso de API `SendCommand` para este libro mayor poner en marcha todos los comandos de PartiQL (por lo tanto, `ALLOW_ALL`) en cualquier tabla del libro mayor especificado. Este modo no tendrá en cuenta las políticas de permisos de IAM de la tabla o comando que cree para el libro mayor.
  - Estándar: (recomendado) un modo de permisos que habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL. Recomendamos encarecidamente usar este modo de permisos para maximizar la seguridad de los datos del libro mayor.

    De forma predeterminada, este modo deniega todas las solicitudes para poner en marcha cualquier comando de PartiQL en tablas de este libro mayor. Para permitir los comandos de PartiQL, debe crear políticas de permisos de IAM para recursos de tablas y acciones de PartiQL específicos, además del permiso de API `SendCommand` para el libro mayor. Para obtener información, consulte Introducción al modo de permisos estándar en Amazon QLDB.
- Protección de eliminación: (opcional) la marca que impide que un usuario elimine un libro mayor. Si no lo especifica al crear el libro mayor, esta característica estará habilitada (`true`) de forma predeterminada.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede deshabilitarlo utilizando la operación `UpdateLedger` para establecer la marca en `false`.

- AWS KMS key: (opcional) la clave de AWS Key Management Service (AWS KMS) que se usará para cifrar los datos en reposo. Elija uno de los siguientes tipos de AWS KMS keys:
  - Clave KMS propiedad de AWS: utilice una clave KMS propiedad de AWS, que también la administra en su nombre.

    Si no define este parámetro durante la creación del libro mayor, se usará este tipo de clave por defecto. También puede usar la cadena `AWS_OWNED_KMS_KEY` para especificar el tipo de clave. Esta opción no requiere ninguna configuración adicional.
  - Clave KMS administrada por el cliente: utilice la clave KMS de cifrado simétrico en la cuenta que ha creado, que posee y que administra. QLDB no admite claves asimétricas.

    Esta opción requiere que cree una clave KMS o use una clave existente en su cuenta. Para obtener instrucciones sobre cómo crear una clave administrada, consulte Creación de claves KMS de cifrado simétrica en la Guía para desarrolladores de AWS Key Management Service.

    Puede especificar una clave KMS administrada por el cliente utilizando un ID, un alias o el Nombre de recurso de Amazon (ARN). Para obtener más información, consulte Identificadores de clave (KeyId) en la Guía para desarrolladores de AWS Key Management Service.

    > **Note**
    >
    > No se admiten claves entre regiones. La clave KMS especificada debe estar en la misma Región de AWS que el libro mayor.

  Para obtener más información, consulte Cifrado en reposo en Amazon QLDB.
- Etiquetas: (opcional) agregue metadatos al libro mayor asociando etiquetas como pares de clave-valor. Puede añadir etiquetas a su libro mayor para ayudar en su organización e identificación. Para obtener más información, consulte Etiquetado de recursos de Amazon QLDB.

El libro mayor no está listo para usarse hasta que QLDB lo haya creado y haya establecido su estado en `ACTIVE`.

Creación de un libro mayor (Java)

Para crear un libro mayor utilizando AWS SDK for Java

1. Cree una instancia de la clase `AmazonQLDB`.
2. Cree una instancia de la clase `CreateLedgerRequest` para proporcionar la información de solicitud.

   Debe proporcionar el nombre del libro mayor y un modo de permisos.
3. Ejecute el método `createLedger` proporcionando el objeto de solicitud como parámetro.

La solicitud `createLedger` devuelve un objeto `CreateLedgerResult` que contiene información sobre el libro mayor. Consulte la siguiente sección para ver un ejemplo del uso de la operación `DescribeLedger` para comprobar el estado del libro mayor después de su creación.

Los siguientes ejemplos demuestran los pasos anteriores.

Example – Use los ajustes de configuración predeterminados

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
CreateLedgerResult result = client.createLedger(request);
```

> **Note**
>
> El libro mayor emplea los siguientes parámetros por defecto si no los especifica:
>
> - Protección de eliminaciones: habilitada (`true`).
> - Clave KMS: clave KMS propiedad de AWS.

Example – Desactive la protección contra la eliminación, use una clave KMS administrada por el cliente y adjunte etiquetas

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();

Map<String, String> tags = new HashMap<>();
```

```
tags.put("IsTest", "true");
tags.put("Domain", "Test");

CreateLedgerRequest request = new CreateLedgerRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD)
        .withDeletionProtection(false)
        .withKmsKey("arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab")
        .withTags(tags);
CreateLedgerResult result = client.createLedger(request);
```

Creación de un libro mayor (AWS CLI)

Cree un nuevo libro mayor denominado `vehicle-registration` usando los parámetros de configuración predeterminados.

Example

```
aws qldb create-ledger --name vehicle-registration --permissions-mode STANDARD
```

> **Note**
>
> El libro mayor emplea los siguientes parámetros por defecto si no los especifica:
>
> - Protección de eliminaciones: habilitada (`true`).
> - Clave KMS: clave KMS propiedad de AWS.

O bien cree un nuevo libro mayor denominado `vehicle-registration` con la protección de eliminación deshabilitada, una clave KMS administrada por el cliente y etiquetas específicas.

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
```

Creación de un libro mayor (AWS CloudFormation)

También puede usar una plantilla de AWS CloudFormation para crear libros mayores. Para obtener más información, consulte el recurso AWS::QLDB::Ledger en la Guía de usuario de AWS CloudFormation.

Descripción de un libro mayor

Utilice la operación `DescribeLedger` para ver información detallada sobre un libro mayor. Debe proporcionar el nombre del libro mayor. El resultado de `DescribeLedger` tiene el mismo formato que el de `CreateLedger`; Contiene la información siguiente:

- Nombre de libro mayor: el nombre del libro mayor que desea describir.
- ARN: el Nombre de recurso de Amazon (ARN) para el libro mayor en el siguiente formato.

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

- Protección contra la eliminación: marca que indica si la característica de protección contra la eliminación está habilitada.
- Fecha y hora de creación: la fecha y la hora, en formato de tiempo epoch, en que se creó el libro mayor.
- Estado: estado actual del libro mayor. Puede ser uno de los siguientes valores:
  - `CREATING`
  - `ACTIVE`
  - `DELETING`
  - `DELETED`
- Modo de permisos: el modo de permisos que se ha asignado al libro mayor. Puede ser uno de los siguientes valores:
  - `ALLOW_ALL`: un modo de permisos heredado que habilita el control de acceso con granularidad a nivel de la API para libros mayores.
  - `STANDARD`: un modo de permisos que habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL.
- Descripción del cifrado: información sobre el cifrado de los datos en reposo del libro mayor. Estas incluyen las siguientes elementos:

- ARN de AWS KMS key: el ARN de la clave KMS administrada por el cliente que emplea el libro mayor para el cifrado en reposo. Si no se ha definido, el libro mayor emplea una clave KMS propiedad de AWS para el cifrado.
- Estado de cifrado: estado actual del cifrado en reposo del libro mayor. Puede ser uno de los siguientes valores:
  - `ENABLED`: el cifrado está totalmente activado con la clave especificada.
  - `UPDATING`: el cambio de la clave especificada se está procesando activamente.

    Los cambios de clave en QLDB son asíncronos. Mientras se procesa el cambio de clave, el registro es totalmente accesible sin ningún impacto en el rendimiento. El tiempo que tarda en actualizarse una clave varía según el tamaño del libro mayor.
  - `KMS_KEY_INACCESSIBLE`: no se puede acceder a la clave KMS administrada por el cliente, y el libro mayor está dañado. La clave se ha deshabilitado o eliminado, o bien se han revocado las concesiones de la clave. Cuando un libro mayor está dañado, no es accesible y no acepta solicitudes de lectura ni escritura.

    Un libro mayor dañado vuelve automáticamente a su estado activo después de restablecer las concesiones de la clave o de volver a activar la clave que estaba desactivada. Sin embargo, la eliminación de una clave KMS administrada por el cliente es irreversible. Una vez que se elimina una clave, ya no puede acceder a los libros mayores que están protegidos con ella y los datos se vuelven irrecuperables permanentemente.
- Inaccesible AWS KMS key: en caso de error, la fecha y hora, en formato de valor de tiempo, en las que la clave KMS se ha vuelto inaccesible por primera vez.

  Si se puede acceder a la clave KMS, este valor no aparece definido.

Para obtener más información, consulte Cifrado en reposo en Amazon QLDB.

> **Note**
>
> Tras crear un libro mayor de QLDB, estará listo para su uso cuando su estado pase de `CREATING` a `ACTIVE`.

Descripción de un libro mayor (Java)

Para describir un libro mayor mediante AWS SDK for Java

1. Cree una instancia de la clase `AmazonQLDB`. También puede usar la misma instancia del cliente de `AmazonQLDB` instanciada para la solicitud `CreateLedger`.
2. Cree una instancia de la clase `DescribeLedgerRequest` y proporcione el nombre del libro mayor que desea describir.
3. Ejecute el método `describeLedger` proporcionando el objeto de solicitud como parámetro.
4. La solicitud `describeLedger` devuelve un objeto `DescribeLedgerResult` que contiene información actual sobre el libro mayor.

En el siguiente ejemplo de código se ponen en práctica los pasos anteriores. Puede llamar al método `describeLedger` del cliente para obtener información sobre el libro mayor en cualquier momento.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
DescribeLedgerRequest request = new DescribeLedgerRequest().withName(ledgerName);
DescribeLedgerResult result = client.describeLedger(request);
System.out.printf("%s: ARN: %s \t State: %s \t CreationDateTime: %s \t
 DeletionProtection: %s
                   \t PermissionsMode: %s \t EncryptionDescription: %s",
        result.getName(),
        result.getArn(),
        result.getState(),
        result.getCreationDateTime(),
        result.getDeletionProtection(),
        result.getPermissionsMode(),
        result.getEncryptionDescription());
```

Descripción de un libro mayor (AWS CLI)

Descripción del libro mayor `vehicle-registration` que acaba de crear.

Example

```
aws qldb describe-ledger --name vehicle-registration
```

Actualización de un libro mayor

Actualmente, la operación `UpdateLedger` permite cambiar los siguientes parámetros de configuración de un libro mayor existente:

- Protección de eliminación: la marca que impide que un usuario elimine un libro mayor. Si esta función está habilitada, deberá deshabilitarla antes configurando el indicador como `false` para poder eliminar el libro mayor.

  Si no define este parámetro, no se efectuarán cambios en la configuración de protección contra eliminación del libro mayor.
- AWS KMS key: la clave de AWS Key Management Service (AWS KMS) que se usará para cifrar los datos en reposo. Si no define este parámetro, no se efectuarán cambios en la clave KMS del libro mayor.

  > **Note**
  >
  > Amazon QLDB lanzó el soporte para AWS KMS keys administrado por el cliente el 22 de julio de 2021. Todos los libros mayores creados antes del lanzamiento están protegidos por Claves propiedad de AWS de forma predeterminada, pero actualmente no es posible cifrar sus datos en reposo con claves administradas por el cliente.
  > Puede ver la hora de creación de su libro mayor en la consola de QLDB.

  Utilice una de las siguientes opciones:

  - Clave KMS propiedad de AWS: utilice una clave KMS propiedad de AWS, que también la administra en su nombre. Para usar este tipo de clave, especifique la cadena `AWS_OWNED_KMS_KEY` de este parámetro. Esta opción no requiere ninguna configuración adicional.
  - Clave KMS administrada por el cliente: utilice la clave KMS de cifrado simétrico en la cuenta que ha creado, que posee y que administra. QLDB no admite claves asimétricas.

    Esta opción requiere que cree una clave KMS o use una clave existente en su cuenta. Para obtener instrucciones sobre cómo crear una clave administrada, consulte Creación de claves KMS de cifrado simétrica en la Guía para desarrolladores de AWS Key Management Service.

Puede especificar una clave KMS administrada por el cliente utilizando un ID, un alias o el Nombre de recurso de Amazon (ARN). Para obtener más información, consulte Identificadores de clave (KeyId) en la Guía para desarrolladores de AWS Key Management Service.

> **Note**
>
> No se admiten claves entre regiones. La clave KMS especificada debe estar en la misma Región de AWS que el libro mayor.

Los cambios de clave en QLDB son asíncronos. Mientras se procesa el cambio de clave, el registro es totalmente accesible sin ningún impacto en el rendimiento.

Puede alternar claves con tanta frecuencia como sea necesario, pero la cantidad de tiempo que tarda en actualizarse una clave varía dependiendo del tamaño del libro mayor. Puede usar la operación `DescribeLedger` para comprobar el estado del cifrado en reposo.

Para obtener más información, consulte Cifrado en reposo en Amazon QLDB.

El resultado de `UpdateLedger` tiene el mismo formato que el de `CreateLedger`;

Actualización de un libro mayor (Java)

Para actualizar un libro mayor utilizando AWS SDK for Java

1. Cree una instancia de la clase `AmazonQLDB`.
2. Cree una instancia de la clase `UpdateLedgerRequest` para proporcionar la información de solicitud.

   Debe proporcionar el nombre del libro mayor junto con un nuevo valor booleano para protegerlo de la eliminación, o bien un nuevo valor de cadena para la clave KMS.
3. Ejecute el método `updateLedger` proporcionando el objeto de solicitud como parámetro.

En el siguiente ejemplo de código se ponen en práctica los pasos anteriores. La solicitud `updateLedger` devuelve un objeto `UpdateLedgerResult` que contiene información actualizada sobre el libro mayor.

Example – Deshabilitar la protección contra la eliminación

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withDeletionProtection(false);
UpdateLedgerResult result = client.updateLedger(request);
```

Example – Utilizar una clave KSM administrada por el cliente.

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab")
UpdateLedgerResult result = client.updateLedger(request);
```

Example – Use una clave KMS propiedad de AWS

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerRequest request = new UpdateLedgerRequest()
        .withName(ledgerName)
        .withKmsKey("AWS_OWNED_KMS_KEY")
UpdateLedgerResult result = client.updateLedger(request);
```

Actualización de un libro mayor (AWS CLI)

Si su libro mayor de `vehicle-registration` tiene la protección contra eliminación habilitada, primero debe desactivarla para poder eliminarlo.

Example

```
aws qldb update-ledger --name vehicle-registration --no-deletion-protection
```

También puede cambiar la configuración de cifrado en reposo del libro mayor para usar una clave KMS gestionada por el cliente.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

O bien puede cambiar la configuración de cifrado en reposo para usar una clave KMS propiedad de AWS.

Example

```
aws qldb update-ledger --name vehicle-registration --kms-key AWS_OWNED_KMS_KEY
```

Actualizar el modo de permisos de un libro mayor

La operación `UpdateLedgerPermissionsMode` permite cambiar el modo de permisos de un libro mayor existente. Elija una de las siguientes opciones:

- Permitir todo: un modo de permisos heredado que habilita el control de acceso con granularidad de la API para los libros mayores.

  Este modo permite a los usuarios que tengan el permiso de API `SendCommand` para este libro mayor poner en marcha todos los comandos de PartiQL (por lo tanto, `ALLOW_ALL`) en cualquier tabla del libro mayor especificado. Este modo no tendrá en cuenta las políticas de permisos de IAM de la tabla o comando que cree para el libro mayor.
- Estándar: (recomendado) un modo de permisos que habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL. Recomendamos encarecidamente usar este modo de permisos para maximizar la seguridad de los datos del libro mayor.

  De forma predeterminada, este modo deniega todas las solicitudes para poner en marcha cualquier comando de PartiQL en tablas de este libro mayor. Para permitir los comandos de PartiQL, debe crear políticas de permisos de IAM para recursos de tablas y acciones de PartiQL específicos, además del permiso de API `SendCommand` para el libro mayor. Para obtener información, consulte Introducción al modo de permisos estándar en Amazon QLDB.

> **Important**
>
> Antes de cambiar al modo de permisos `STANDARD`, deberá crear todas las políticas de IAM y etiquetas de tabla necesarias para evitar interrupciones en el trabajo de los usuarios. Para obtener más información, consulte Migrar al modo de permisos estándar.

Actualización del modo de permisos de un libro mayor (Java)

Para actualizar el modo de permisos de un libro mayor mediante AWS SDK for Java

1. Cree una instancia de la clase `AmazonQLDB`.
2. Cree una instancia de la clase `UpdateLedgerPermissionsModeRequest` para proporcionar la información de solicitud.

   Debe proporcionar el nombre del libro mayor junto con un nuevo valor de cadena para el modo de permisos.
3. Ejecute el método `updateLedgerPermissionsMode` proporcionando el objeto de solicitud como parámetro.

En el siguiente ejemplo de código se ponen en práctica los pasos anteriores. La solicitud `updateLedgerPermissionsMode` devuelve un objeto `UpdateLedgerPermissionsModeResult` que contiene información actualizada sobre el libro mayor.

Example – Asigne el modo de permisos estándar

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
UpdateLedgerPermissionsModeRequest request = new UpdateLedgerPermissionsModeRequest()
        .withName(ledgerName)
        .withPermissionsMode(PermissionsMode.STANDARD);
UpdateLedgerPermissionsModeResult result = client.updateLedgerPermissionsMode(request);
```

Actualización del modo de permisos de un libro mayor (AWS CLI)

Asigne el modo de permisos `STANDARD` a su libro mayor `vehicle-registration`.

Example

```
aws qldb update-ledger-permissions-mode --name vehicle-registration --permissions-mode
 STANDARD
```

Migrar al modo de permisos estándar

Para migrar al modo de permisos `STANDARD`, le recomendamos analizar sus patrones de acceso a QLDB y añadir políticas de IAM que concedan a los usuarios los permisos adecuados para acceder a sus recursos.

Antes de cambiar al modo de permisos `STANDARD`, deberá crear todas las políticas de IAM y etiquetas de tabla necesarias. De lo contrario, el cambio del modo de permisos podría interrumpir el trabajo de los usuarios hasta que cree las políticas de IAM correctas o restablezca el modo de permisos a `ALLOW_ALL`. Para obtener información sobre la creación de estas políticas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

También puede utilizar la política administrada de AWS para conceder acceso completo a todos los recursos de QLDB. Las políticas gestionadas `AmazonQLDBFullAccess` y `AmazonQLDBConsoleFullAccess` incluyen todas las acciones de QLDB, incluidas todas las acciones de PartiQL. Adjuntar una de estas políticas a una entidad principal equivale a conceder el modo de permisos `ALLOW_ALL` a esa entidad principal. Para obtener más información, consulte AWS políticas gestionadas para Amazon QLDB.

Eliminación de un libro mayor

Use la operación `DeleteLedger` para eliminar un libro mayor y todo su contenido. La eliminación de un libro mayor es una operación irrecuperable.

Si la protección contra eliminación está habilitada para su libro mayor, primero debe desactivarla para poder eliminar el libro mayor.

Cuando se emite una solicitud `DeleteLedger`, el estado del libro mayor cambia de `ACTIVE` a `DELETING`. La eliminación del libro mayor puede tardar un tiempo, en función de la cantidad de almacenamiento utilizado. Cuando la operación `DeleteLedger` concluya, el libro mayor ya no existirá en QLDB.

Eliminación de un libro mayor (Java)

Para eliminar un libro mayor utilizando AWS SDK for Java

1. Cree una instancia de la clase `AmazonQLDB`.
2. Cree una instancia de la clase `DeleteLedgerRequest` y proporcione el nombre del libro mayor que desea eliminar.
3. Ejecute el método `deleteLedger` proporcionando el objeto de solicitud como parámetro.

En el siguiente ejemplo de código se ponen en práctica los pasos anteriores.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
```

```
DeleteLedgerRequest request = new DeleteLedgerRequest().withName(ledgerName);
DeleteLedgerResult result = client.deleteLedger(request);
```

Eliminación de un libro mayor (AWS CLI)

Elimine su libro mayor de `vehicle-registration`.

Example

```
aws qldb delete-ledger --name vehicle-registration
```

Enumerar libros mayores

La operación `ListLedgers` devuelve información resumida de todos los libros de QLDB en la región y Cuenta de AWS actuales.

Enumerar libros mayores (Java)

Para enumerar los libros mayores de su cuenta usando AWS SDK for Java

1. Cree una instancia de la clase `AmazonQLDB`.
2. Cree una instancia de la clase `ListLedgersRequest`.

   Si ha recibido un valor de `NextToken` en la respuesta de una llamada `ListLedgers` previa, debe proporcionarlo en esta solicitud para pasar a la siguiente página de resultados.
3. Ejecute el método `listLedgers` proporcionando el objeto de solicitud como parámetro.
4. La solicitud `listLedgers` devuelve un objeto `ListLedgersResult`. Este objeto tiene una lista de objetos `LedgerSummary` y un token de paginación que indica si hay más resultados disponibles:

   - Si `NextToken` está vacío, se ha procesado la última página de resultados y no hay más resultados.
   - Si `NextToken` no está vacío, hay más resultados disponibles. Para recuperar la siguiente página de resultados, use el valor de `NextToken` en una llamada posterior a `ListLedgers`.

En el siguiente ejemplo de código se ponen en práctica los pasos anteriores.

Example

```
AmazonQLDB client = AmazonQLDBClientBuilder.standard().build();
List<LedgerSummary> ledgerSummaries = new ArrayList<>();
String nextToken = null;
do {
    ListLedgersRequest request = new ListLedgersRequest().withNextToken(nextToken);
    ListLedgersResult result = client.listLedgers(request);
    ledgerSummaries.addAll(result.getLedgers());
    nextToken = result.getNextToken();
} while (nextToken != null);
```

Enumeración de libros mayores (AWS CLI)

Enumere todos los libros mayores en la región y Cuenta de AWS actuales.

Example

```
aws qldb list-ledgers
```

Crear recursos de Amazon QLDB con AWS CloudFormation

Amazon QLDB está integrado con AWS CloudFormation, un servicio que lo ayuda a modelar y configurar los recursos de AWS para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Puede crear una plantilla que describa todos los recursos de AWS que desea (como libros mayores de QLDB) y AWS CloudFormation aprovisionará y configurará estos recursos por usted.

Cuando utiliza AWS CloudFormation, puede volver a utilizar la plantilla para configurar sus recursos QLDB de forma coherente y repetida. Solo tiene que describir los recursos una vez y luego aprovisionar los mismos recursos una y otra vez en varias Cuentas de AWS y regiones.

QLDB y plantillas AWS CloudFormation

Para aprovisionar y configurar los recursos de QLDB y sus servicios relacionados, debe entender las plantillas de AWS CloudFormation. Las plantillas son archivos de texto con formato de tipo JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas de AWS CloudFormation. Si no está familiarizado con JSON o YAML, puede utilizar Designer de AWS

CloudFormation para comenzar a utilizar las plantillas de AWS CloudFormation. Para obtener más información, consulte ¿Qué es Designer de AWS CloudFormation? en la Guía del usuario de AWS CloudFormation.

QLDB admite la creación de libros de mayores y secuencias de diarios en AWS CloudFormation. Para obtener más información, incluidos ejemplos de plantillas JSON y YAML para los libros mayores y las secuencias, consulte las referencias siguientes del tipo de recurso en la Guía del usuario de AWS CloudFormation:

- AWS::QLDB::Ledger reference
- AWS::QLDB::Stream reference

Obtener más información sobre AWS CloudFormation

Para obtener más información acerca de AWS CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- Guía del usuario de AWS CloudFormation
- Referencia de la API de AWS CloudFormation
- Guía del usuario de la interfaz de la línea de comandos de AWS CloudFormation

Etiquetado de recursos de Amazon QLDB

Una etiqueta es un atributo personalizado que usted o AWS asignan a un recurso de AWS. Cada etiqueta tiene dos partes:

- Una clave de etiqueta (por ejemplo, `CostCenter`, `Environment` o `Project`). Las claves de etiqueta distinguen entre mayúsculas y minúsculas.
- Un campo opcional denominado valor de etiqueta (por ejemplo, `111122223333` o `Production`). Omitir el valor de etiqueta es lo mismo que utilizar una cadena vacía. Al igual que las claves de etiqueta, los valores de etiqueta distinguen entre mayúsculas y minúsculas.

Las etiquetas le ayudan a hacer lo siguiente:

- Identificar y organizar sus recursos de AWS. Muchos Servicios de AWS admiten el etiquetado, por lo que puede asignar la misma etiqueta a los recursos de diferentes servicios para indicar que los

recursos están relacionados. Por ejemplo, puede asignar la misma etiqueta a un libro mayor de Amazon QLDB que se asigna a un bucket de Amazon S3.

- Realizar un seguimiento de los costos de AWS. Estas etiquetas se activan en el panel de AWS Billing and Cost Management. AWS usa las etiquetas para clasificar los costos y enviar un informe mensual de asignación de costos. Para obtener más información, consulte Uso de etiquetas de asignación de costos en la Guía del usuario de AWS Billing.
- Controle el acceso a los recursos de AWS con AWS Identity and Access Management (IAM). Para obtener más información, consulte Control de acceso basado en atributos (ABAC) con QLDB en esta guía para desarrolladores, y control de acceso mediante etiquetas de IAM en la Guía del usuario de IAM.

Para obtener sugerencias acerca del uso de etiquetas, consulte la publicación sobre Estrategias de etiquetado de AWS en el blog de respuestas de AWS.

En las siguientes secciones, se ofrece más información sobre las etiquetas de Amazon QLDB.

Temas

- Recursos admitidos en Amazon QLDB
- Convenciones de nomenclatura y uso de las etiquetas
- Administración de etiquetas
- Etiquetado de recursos durante la creación

Recursos admitidos en Amazon QLDB

Los siguientes recursos de Amazon QLDB admiten el etiquetado:

- libro mayor
- tabla
- secuencia del diario

Para obtener información acerca de cómo añadir y administrar etiquetas, consulte Administración de etiquetas.

Convenciones de nomenclatura y uso de las etiquetas

Las siguientes convenciones básicas de nomenclatura y uso se aplican al uso de etiquetas con recursos de Amazon QLDB:

- Cada recurso puede tener un máximo de 50 etiquetas.
- Para cada recurso, cada clave de etiqueta debe ser única y solo puede tener un valor.
- La longitud máxima de la clave de etiqueta es de 128 caracteres Unicode en UTF-8.
- La longitud máxima del valor de etiqueta es de 256 caracteres Unicode en UTF-8.
- Los caracteres permitidos son letras, números y espacios representables en UTF-8, además de los siguientes caracteres: . : + = @ _ / - (guion).
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas. Como práctica recomendada, decida una estrategia de uso de mayúsculas y minúsculas en las etiquetas e implemente esa estrategia sistemáticamente en todos los tipos de recursos. Por ejemplo, decida si se va a utilizar `Costcenter`, `costcenter` o `CostCenter` y utilice la misma convención para todas las etiquetas. Procure no utilizar etiquetas similares con un tratamiento de mayúsculas y minúsculas incoherente.
- El prefijo `aws:` se reserva para uso de AWS. No puede editar ni eliminar la clave o el valor de una etiqueta cuando la etiqueta tiene una clave de etiqueta con el prefijo `aws:`. Las etiquetas que tengan este prefijo no cuentan para el límite de etiquetas por recurso.

Administración de etiquetas

Las etiquetas se componen de las propiedades `Key` y `Value` de un recurso. Puede usar la consola de Amazon QLDB, la AWS CLI o la API de QLDB para agregar, editar o eliminar los valores de estas propiedades. También puede utilizar el editor de etiquetas de AWS Resource Groups para administrar etiquetas.

Para obtener información sobre cómo trabajar con etiquetas, consulte las siguientes operaciones de la API:

- ListTagsForResource en la Referencia de la API de Amazon QLDB
- TagResource en la Referencia de la API de Amazon QLDB
- UntagResource en la Referencia de la API de Amazon QLDB

Para utilizar el panel de etiquetado QLDB (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon QLDB en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, seleccione Libros mayores.
3. En la lista de libros mayores, seleccione el nombre del libro mayor cuyas etiquetas desee administrar.
4. En la página de detalles del libro mayor, localice la tarjeta Etiquetas y seleccione Administrar etiquetas.
5. En la página Administrar etiquetas, puede añadir, editar o eliminar cualquier etiqueta que considere adecuada para su libro mayor. Una vez que las claves y valores de etiqueta sean los deseados, seleccione Guardar.

Etiquetado de recursos durante la creación

Para los recursos de QLDB que admiten el etiquetado, puede definir etiquetas mientras crea el recurso mediante la AWS Management Console, la AWS CLI o la API de QLDB. Al etiquetar los recursos en el momento de su creación, ya no es necesario ejecutar scripts de etiquetado personalizados después de la creación del recurso.

Una vez etiquetada una tabla, puede controlar el acceso a la tabla en función de esas etiquetas. Por ejemplo, puede conceder acceso total solo a los recursos de la tabla que tengan una etiqueta específica. Para ver una política de ejemplo JSON, consulte Acceso completo a todas las acciones basadas en las etiquetas de las tablas.

> **Note**
>
> Los recursos de secuencia no heredan las etiquetas de su recurso de libro mayor raíz.

También puede definir etiquetas de tabla especificándolas en una instrucción `CREATE TABLE` de PartiQL. Para obtener más información, consulte Etiquetar tablas.

Seguridad en Amazon QLDB

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que se ejecuta Servicios de AWS en la Nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores independientes prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información sobre los programas de conformidad que se aplican a Amazon QLDB, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por lo Servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos vigentes.

Esta documentación lo ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza QLDB. En los siguientes temas, se le mostrará cómo configurar QLDB para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a usar otros Servicios de AWS que le ayuden a monitorear y proteger sus recursos de QLDB.

Temas

Protección de los datos en Amazon QLDB

El modelo de se aplica a protección de datos en Amazon QLDB. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los. Nube de AWS Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con QLDB u Servicios de AWS otro mediante la consola, la API AWS CLI o los SDK. AWS Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación

o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

> **Note**
>
> Esta guía sobre cómo evitar etiquetas o campos de formato libre para información confidencial se refiere a los metadatos de un recurso del libro mayor de QLDB, en lugar de a los datos almacenados en el libro mayor. Los datos almacenados en un recurso contable de QLDB no se utilizan para los registros de facturación o diagnóstico.

Temas

Cifrado en reposo en Amazon QLDB

Todos los datos de usuario almacenados en Amazon QLDB están completamente cifrados en reposo. El cifrado QLDB en reposo proporciona una seguridad mejorada al cifrar todos los datos del libro mayor en reposo mediante claves de cifrado en (). AWS Key Management Service AWS KMS Esta funcionalidad ayuda a reducir la carga y la complejidad operativas que conlleva la protección de información confidencial. Con el cifrado en reposo, puede crear aplicaciones de libro mayor sensibles a la seguridad que necesitan cumplimiento estricto de cifrado y requisitos normativos.

El cifrado en reposo se integra AWS KMS para administrar la clave de cifrado que se utiliza para proteger sus libros de contabilidad QLDB. Para obtener más información al respecto AWS KMS, consulte AWS Key Management Service los conceptos de la AWS Key Management Service Guía para desarrolladores.

En QLDB, puede especificar el tipo de recurso de registro para cada recurso AWS KMS key contable. Al crear un nuevo libro mayor o actualizar uno existente, puede elegir uno de los siguientes tipos de claves KMS para proteger los datos del libro mayor:

- Clave propiedad de AWS: tipo de cifrado predeterminado. La clave es propiedad de QLDB (sin cargo adicional).
- Clave administrada por el cliente: la clave se almacena en su Cuenta de AWS y usted la crea, posee y administra. Usted tiene el control total sobre la clave (se aplican AWS KMS cargos).

> **Note**
>
> Amazon QLDB lanzó el soporte para la AWS KMS keys gestión de clientes el 22 de julio de 2021. Todos los libros de contabilidad que se hayan creado antes del lanzamiento están protegidos de forma Claves propiedad de AWS predeterminada, pero actualmente no son aptos para el cifrado en reposo con claves administradas por el cliente.
> Puede ver la hora de creación de su libro mayor en la consola de QLDB.

Cuando obtiene acceso al libro mayor, QLDB descifra los datos de forma transparente. Puede cambiar entre la clave gestionada por el cliente Clave propiedad de AWS y la clave gestionada por el cliente en cualquier momento. No es necesario que cambie sus aplicaciones o código para utilizar o administrar datos cifrados.

Puede especificar una clave de cifrado al crear un nuevo libro mayor o cambiar la clave de cifrado de un libro mayor existente mediante la AWS Management Console API de QLDB o (). AWS Command Line Interface AWS CLI Para obtener más información, consulte Uso de claves administradas por el cliente en Amazon QLDB.

> **Note**
>
> De forma predeterminada, Amazon QLDB habilita automáticamente el cifrado en reposo con Claves propiedad de AWS sin costo adicional. Sin embargo, se aplican AWS KMS cargos por el uso de una clave administrada por el cliente. Para obtener información sobre precios, consulte Precios de AWS Key Management Service.
> El cifrado QLDB en reposo está disponible en todos los Regiones de AWS lugares donde esté disponible QLDB.

Temas

Cifrado en reposo: cómo funciona en Amazon QLDB

El cifrado en reposo de QLDB cifra sus datos mediante el estándar de cifrado avanzado de 256 bits (AES-256). Esto ayuda a proteger los datos del acceso no autorizado al almacenamiento

subyacente. De manera predeterminada, todos los datos almacenados en los libros mayores de QLDB se cifran en reposo. El cifrado del servidor es transparente, lo que significa que no es necesario realizar cambios en las aplicaciones.

Encryption at rest se integra con AWS Key Management Service (AWS KMS) para administrar la clave de cifrado que se utiliza para proteger sus libros de contabilidad QLDB. Al crear un libro mayor nuevo o actualizar un libro mayor existente, puede elegir uno de los siguientes tipos de claves AWS KMS :

- Clave propiedad de AWS: tipo de cifrado predeterminado. La clave es propiedad de QLDB (sin cargo adicional).
- Clave administrada por el cliente: la clave se almacena en su Cuenta de AWS y usted la crea, posee y administra. Usted tiene el control total sobre la clave (de AWS KMS pago).

Temas

Clave propiedad de AWS

Claves propiedad de AWS no están almacenados en su Cuenta de AWS. Forman parte de una colección de claves de KMS que AWS posee y administra para su uso en múltiples direcciones Cuentas de AWS. Servicios de AWS se pueden utilizar Claves propiedad de AWS para proteger sus datos.

No es necesario crear ni administrar las Claves propiedad de AWS. Sin embargo, no puede ver Claves propiedad de AWS, rastrear ni auditar su uso. No se te cobra una cuota mensual ni una cuota de uso Claves propiedad de AWS, y estas no se tienen en cuenta para las AWS KMS cuotas de tu cuenta.

Para obtener más información, consulte Claves propiedad de AWS en la Guía para desarrolladores de AWS Key Management Service .

Clave administrada por clientes

Las claves administradas por el cliente son claves de KMS Cuenta de AWS que usted crea, posee y administra. Usted tiene el control total sobre estas claves de KMS. QLDB solo es compatible con claves de cifrado de KMS simétricas.

Utilice una clave administrada por el cliente para obtener las siguientes características:

- Configuración y mantenimiento de las políticas de claves, políticas de IAM y concesiones para controlar el acceso a las claves de KMS
- Activar y desactivar la clave
- Rotar el material criptográfico para la clave
- Crear etiquetas clave y alias
- Programar la clave para eliminarla
- Importar su propio material de claves o usar un almacén de claves personalizadas de su propiedad y que administra
- Uso AWS CloudTrail de Amazon CloudWatch Logs para rastrear las solicitudes que QLDB envía AWS KMS en su nombre

Para más información, consulte las claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service .

Las claves administradas por el cliente conllevan un cargo por cada llamada a la API y se aplican AWS KMS cuotas a estas claves de KMS. Para obtener más información, consulte cuotas de solicitudes o de recursos de AWS KMS.

Cuando se especifica una clave administrada por el cliente como clave KMS para un libro mayor, todos los datos del libro mayor, tanto del almacenamiento del diario como del almacenamiento indexado, se protegen con la misma clave administrada por el cliente.

Claves administradas por el cliente inaccesibles

Si desactiva la clave administrada por el cliente, programa su eliminación o revoca las concesiones de la clave, el estado del cifrado de su libro mayor pasa a ser `KMS_KEY_INACCESSIBLE`. En este estado, el libro mayor está dañado y no acepta solicitudes de lectura o escritura. Una clave inaccesible impide que todos los usuarios y el servicio QLDB cifren o descifren los datos y realicen operaciones de lectura y escritura en el libro mayor. QLDB debe tener acceso a la clave KMS para asegurarse de que pueda seguir accediendo al libro mayor y evitar la pérdida de datos.

> **Important**
>
> Un libro mayor dañado vuelve automáticamente a su estado activo después de restablecer las concesiones de la clave o de volver a activar la clave que estaba desactivada.
> Sin embargo, eliminar una clave administrada por el cliente es irreversible. Una vez que se elimina una clave, ya no puede acceder a los libros mayores que están protegidos con ella y los datos se vuelven irrecuperables permanentemente.

Para comprobar el estado de cifrado de un libro mayor, utilice la operación AWS Management Console o la DescribeLedgerAPI.

Cómo utiliza Amazon QLDB las concesiones en AWS KMS

QLDB requiere concesiones para utilizar su clave administrada por el cliente. Cuando crea un libro mayor protegido con una clave gestionada por el cliente, la QLDB crea subvenciones en su nombre enviando solicitudes a. CreateGrant AWS KMS Las concesiones entrantes AWS KMS se utilizan para dar acceso a la QLDB a una clave de KMS de un cliente. Cuenta de AWS Para obtener más información, consulte Uso de concesiones en la Guía para desarrolladores de AWS Key Management Service .

QLDB necesita la concesión para utilizar la clave administrada por el cliente para las siguientes operaciones AWS KMS :

- DescribeKey— Compruebe que la clave KMS de cifrado simétrico especificada sea válida.
- GenerateDataKey— Genere una clave de datos simétrica única que QLDB utilice para cifrar los datos en reposo en su libro mayor.
- Decrypt: descifra los datos cifrados con la clave administrada por el cliente.
- Encrypt: cifra el texto sin formato como texto cifrado con la clave administrada por el cliente.

Puede revocar la concesión para eliminar el acceso del servicio a la clave administrada por el cliente en cualquier momento. Si lo hace, la clave se vuelve inaccesible y QLDB pierde el acceso a cualquiera de los datos del libro mayor protegidos por la clave administrada por el cliente. En este estado, el libro mayor está dañado y no acepta solicitudes de lectura o escritura hasta que restablezca las concesiones de la clave.

Restablecer las concesiones en AWS KMS

Para restaurar las concesiones en una clave administrada por el cliente y recuperar el acceso a un libro mayor en QLDB, puede actualizar el libro mayor y especificar la misma clave de KMS. Para ver instrucciones, consulte Actualización de la AWS KMS key de un libro mayor existente.

Consideraciones sobre el cifrado en reposo

Es importante tener en cuenta lo siguiente cuando use el cifrado en reposo en QLDB:

- El cifrado en reposo del servidor está habilitado para los datos de todos los libros mayores de QLDB y no se puede deshabilitar. No puede cifrar solo un subconjunto de elementos de un libro mayor.
- El cifrado en reposo solo cifra los datos mientras están estáticos (en reposo) en un medio de almacenamiento persistente. Si le preocupa la seguridad de los datos cuando están en tránsito o en uso, puede ser conveniente adoptar medidas adicionales como se muestra a continuación:
  - Datos en tránsito: todos los datos de QLDB se cifran en tránsito. De forma predeterminada, las comunicaciones de entrada y salida de QLDB usan el protocolo HTTPS, que protege el tráfico de la red mediante el uso del cifrado de capa de conexión segura (SSL)/seguridad de la capa de transporte (TLS).
  - Datos en uso: proteja lo datos antes de enviarlos a QLDB mediante el cifrado del cliente.

Para obtener información sobre cómo implementar claves administradas por el cliente en los libros mayores, vaya a Uso de claves administradas por el cliente en Amazon QLDB.

Uso de claves administradas por el cliente en Amazon QLDB

Puede utilizar la API AWS Management Console, AWS Command Line Interface (AWS CLI) o QLDB para especificar los libros de contabilidad nuevos y AWS KMS key los libros de contabilidad existentes en Amazon QLDB. Los siguientes temas describen cómo administrar y monitorizar el uso de las claves administradas por el cliente en QLDB.

Temas

Requisitos previos

Antes de poder proteger un libro mayor de QLDB con una clave gestionada por el cliente, primero debe crear la clave en (). AWS Key Management Service AWS KMS También debe especificar una política clave que permita a la QLDB crear subvenciones en su nombre AWS KMS key .

Crear una clave administrada por el cliente

Para crear una clave administrada por el cliente, siga los pasos que se indican en la sección Creating symmetric encryption KMS keys en la Guía para desarrolladores de AWS Key Management Service . QLDB no admite claves asimétricas.

Configurar una política de claves

Las políticas clave son la principal forma de controlar el acceso a las claves gestionadas por los clientes. AWS KMS Cada clave administrada por el cliente debe tener exclusivamente una política de clave. Las declaraciones en el documento de políticas de claves determinan quién tiene permiso para usar la clave KMS y cómo debe usar dicho permiso. Para obtener más información, consulte Uso de políticas clave en AWS KMS.

Cuando crea la clave administrada por el cliente, puede especificar una política de claves. Para cambiar una política de claves por una clave administrada por el cliente existente, consulte Cambiar una política de claves.

Para permitir que QLDB utilice su clave gestionada por el cliente, la política de claves debe incluir permisos para las siguientes acciones: AWS KMS

- kms: CreateGrant — Añade una concesión a una clave gestionada por el cliente. Otorga acceso de control a una clave KMS específica.

  Al crear o actualizar un libro mayor con una clave administrada por el cliente específica, QLDB crea concesiones que permiten el acceso a las operaciones de concesión que requiere. Las operaciones de concesiones incluyen lo siguiente:

  - GenerateDataKey
  - Decrypt
  - Encrypt
- kms: DescribeKey — Devuelve información detallada sobre una clave gestionada por el cliente. QLDB utiliza esta información para validar la clave.

Política de claves de ejemplo

A continuación, se muestra una política de claves de ejemplo que puede utilizar para QLDB. Esta política permite a entidades principales que están autorizadas a utilizar QLDB desde la `111122223333` cuenta realizar llamadas a las operaciones `DescribeKey` y `CreateGrant` del recurso `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`.

Para usar esta política, sustituya *`us-east-1`*, *`111122223333`* y *`1234abcd-12ab-34cd-56ef-1234567890ab`* del ejemplo por su propia información.

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

Especificar una AWS KMS key para un nuevo libro mayor

Siga estos pasos para especificar una clave KMS al crear un nuevo libro mayor mediante la consola de QLDB o la AWS CLI.

Puede especificar una clave administrada por el cliente utilizando un ID, un alias o el Nombre de recurso de Amazon (ARN). Para obtener más información, consulte los identificadores clave (KeyId) en la Guía para AWS Key Management Service desarrolladores.

> **Note**
>
> No se admiten claves entre regiones. La clave KMS especificada debe estar en el mismo lugar que Región de AWS el libro mayor.

Creación de un libro mayor (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. Elija Crear libro mayor.
3. En la página Crear libro mayor, proceda del modo siguiente:
   - Información del libro mayor: introduzca un nombre de libro mayor que sea único entre todos los libros de contabilidad actuales y de la región. Cuenta de AWS
   - Modo de permisos: el modo de permisos que se va a asignar al libro mayor:
     - Permitir todo
     - Estándar (recomendado)
   - Cifrar datos en reposo: elija el tipo de clave KMS que se debe usarse para el cifrado en reposo:
     - Utilice una clave AWS de KMS propia: utilice una clave de KMS que sea de su propiedad y que administre AWS en su nombre. Esta es la opción predeterminada y no requiere ninguna configuración adicional.
     - Elige una AWS KMS clave diferente: usa una clave KMS de cifrado simétrico en tu cuenta que crees, poseas y administres.

       Para crear una clave nueva mediante la AWS KMS consola, selecciona Crear una AWS KMS clave. Para más información, consulte Creación de claves de KMS de cifrado simétricas en la Guía para desarrolladores de AWS Key Management Service .

       Para usar una clave de KMS existente, elija una de la lista desplegable o especifique un ARN de clave de KMS.
4. Cuando la configuración sea la deseada, elija Crear libro mayor.

Puede acceder a su libro mayor de QLDB cuando su estado pase a ser Activo. Esto puede tardar varios minutos.

Creación de un libro mayor (AWS CLI)

Úselo AWS CLI para crear un libro mayor en QLDB con la clave Clave propiedad de AWS predeterminada o gestionada por el cliente.

Example — Creación de un libro mayor con la Clave propiedad de AWS predeterminada

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

Example — Creación de un libro mayor con una clave administrada por el cliente

```
aws qldb create-ledger \
    --name my-example-ledger \
    --permissions-mode STANDARD \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Actualización de la AWS KMS key de un libro mayor existente

También puede utilizar la consola QLDB o la para actualizar AWS CLI la clave KMS de un libro mayor existente a una clave gestionada por el cliente o a Clave propiedad de AWS una clave gestionada por el cliente en cualquier momento.

> **Note**
>
> Amazon QLDB lanzó el soporte para la AWS KMS keys gestión de clientes el 22 de julio de 2021. Todos los libros de contabilidad que se hayan creado antes del lanzamiento están protegidos de forma Claves propiedad de AWS predeterminada, pero actualmente no son aptos para el cifrado en reposo con claves administradas por el cliente.
> Puede ver la hora de creación de su libro mayor en la consola de QLDB.

Los cambios de clave en QLDB son asíncronos. Mientras se procesa el cambio de clave, el registro es totalmente accesible sin ningún impacto en el rendimiento. El tiempo que tarda en actualizarse una clave varía según el tamaño del libro mayor.

Puede especificar una clave administrada por el cliente utilizando un ID, un alias o el Nombre de recurso de Amazon (ARN). Para obtener más información, consulte los identificadores de clave (KeyId) en la Guía AWS Key Management Service para desarrolladores.

> **Note**
>
> No se admiten claves entre regiones. La clave KMS especificada debe estar en el mismo lugar que Región de AWS el libro mayor.

Actualización de un libro mayor (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Libros mayores.
3. En la lista de libros mayores, seleccione el libro mayor que desee actualizar y, a continuación, elija Editar libro mayor.
4. En la página Editar libro mayor, elija el tipo de clave KMS que se utilizará para el cifrado en reposo:

   - Utilice una clave AWS de KMS propia: utilice una clave de KMS que sea de su propiedad y que esté gestionada AWS por usted. Esta es la opción predeterminada y no requiere ninguna configuración adicional.
   - Elige una AWS KMS clave diferente: usa una clave KMS de cifrado simétrico en tu cuenta que crees, poseas y administres.

     Para crear una clave nueva mediante la AWS KMS consola, selecciona Crear una AWS KMS clave. Para más información, consulte Creación de claves de KMS de cifrado simétricas en la Guía para desarrolladores de AWS Key Management Service .

     Para usar una clave de KMS existente, elija una de la lista desplegable o especifique un ARN de clave de KMS.
5. Seleccione Confirmar cambios.

Actualización de un libro mayor (AWS CLI)

Úselo AWS CLI para actualizar un libro mayor existente en QLDB con la clave Clave propiedad de AWS predeterminada o gestionada por el cliente.

Example — Actualización de un libro mayor con la Clave propiedad de AWS predeterminada

```
aws qldb update-ledger --name my-example-ledger --kms-key AWS_OWNED_KMS_KEY
```

Example — Actualización de un libro mayor con una clave administrada por el cliente

```
aws qldb update-ledger \
    --name my-example-ledger \
    --kms-key arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Monitoreo de AWS KMS keys

Si utilizas una clave gestionada por el cliente para proteger tus libros de contabilidad de Amazon QLDB, puedes utilizar AWS CloudTrail CloudWatch Amazon Logs para realizar un seguimiento de las solicitudes que QLDB envía en tu nombre. AWS KMS Para obtener más información, consulte Monitorear AWS KMS keys en la Guía para desarrolladores de AWS Key Management Service .

Los siguientes ejemplos son entradas de CloudTrail registro para las operaciones`CreateGrant`,,, y`GenerateDataKey`. `Decrypt` `Encrypt` `DescribeKey`

CreateGrant

Cuando especifica una clave gestionada por el cliente para proteger su libro mayor, QLDB `CreateGrant` envía solicitudes en su nombre para permitir el acceso AWS KMS a su clave KMS. Además, QLDB utiliza la operación `RetireGrant` para eliminar las concesiones cuando elimina un libro mayor.

Las concesiones que crea QLDB son específicas de un libro mayor. La entidad principal en la solicitud `CreateGrant` es el usuario que creó la tabla.

El evento que registra la operación `CreateGrant` es similar al siguiente evento de ejemplo. Los parámetros incluyen el nombre de recurso de Amazon (ARN) de la clave administrada por el cliente, la entidad principal beneficiaria, la entidad principal que retira la concesión (el servicio de QLDB) y las operaciones que cubre la concesión.

```
{
    "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

Cuando especifica una clave administrada por el cliente para proteger su libro mayor, QLDB crea una clave de datos única. Envía una `GenerateDataKey` solicitud a la AWS KMS que se especifica la clave gestionada por el cliente para el libro mayor.

El evento que registra la operación `GenerateDataKey` es similar al siguiente evento de ejemplo. El usuario es la cuenta del servicio QLDB. Los parámetros incluyen el ARN de la clave administrada por el cliente, un especificador de clave de datos que requiere una longitud de 32 bytes y el contexto de cifrado que identifica el nodo interno de la jerarquía de claves.

```
{
    "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

Al acceder a un libro mayor, QLDB llama a la operación `Decrypt` para descifrar la clave de datos almacenada del libro mayor para que pueda acceder a los datos cifrados del libro mayor.

El evento que registra la operación `Decrypt` es similar al siguiente evento de ejemplo. El usuario es la cuenta del servicio QLDB. Los parámetros incluyen el ARN de la clave administrada por el cliente y el contexto de cifrado que identifica el nodo interno de la jerarquía de claves.

```
{
    "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 llama a la operación `Encrypt` para cifrar texto sin formato en texto cifrado mediante la clave administrada por el cliente.

El evento que registra la operación `Encrypt` es similar al siguiente evento de ejemplo. El usuario es la cuenta del servicio QLDB. Los parámetros incluyen el ARN de la clave administrada por el cliente y el contexto de cifrado que especifica el identificador único interno del libro mayor.

```
{
    "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 llama a la operación `DescribeKey` para determinar si la clave KMS que ha especificado existe en Cuenta de AWS y en la región.

El evento que registra la operación `DescribeKey` es similar al siguiente evento de ejemplo. El principal es el usuario suyo Cuenta de AWS que especificó la clave KMS. Los parámetros incluyen el ARN de la clave administrada por el cliente.

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:sample-user",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/sample-user",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-06-04T21:37:11Z"
            }
        },
        "invokedBy": "qldb.amazonaws.com"
    },
    "eventTime": "2021-06-04T21:40:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "qldb.amazonaws.com",
    "userAgent": "qldb.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "a30586af-c783-4d25-8fda-33152c816c36",
    "eventID": "7a9caf07-2b27-44ab-afe4-b259533ebb88",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

Cifrado en tránsito en Amazon QLDB

Amazon QLDB solo acepta conexiones seguras que usen el protocolo HTTPS, que protege el tráfico de la red mediante el uso de Capa de conexión segura (SSL)/seguridad de la capa de transporte (TLS). El cifrado en tránsito proporciona una capa adicional de protección de datos al cifrarlos mientras viajan hacia y desde QLDB. Las políticas de la organización, las normativas industriales o gubernamentales y los requisitos de conformidad suelen requerir el uso del cifrado en tránsito para aumentar la seguridad de los datos de las aplicaciones cuando transmiten datos a la red.

QLDB también ofrece puntos de conexión FIPS en regiones seleccionadas. A diferencia de los puntos de conexión estándar de AWS , los puntos de conexión FIPS utilizan una biblioteca de software TLS que cumple los Estándares Federales de Procesamiento de la Información (FIPS) 140-2. Las empresas que trabajan con el gobierno de los Estados Unidos pueden requerir estos puntos de conexión. Para obtener más información, consulte los puntos de conexión FIPS en Referencia general de AWS. Para obtener una lista de las regiones y puntos de conexión disponibles para QLDB, consulte Puntos de conexión y cuotas de Amazon QLDB.

Administración de identidades y accesos para Amazon QLDB

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién puede estar autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de QLDB. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración de acceso mediante políticas

- Cómo funciona Amazon QLDB con IAM
- Introducción al modo de permisos estándar en Amazon QLDB
- Ejemplos de políticas basadas en identidades para Amazon QLDB
- Prevención de la sustitución confusa entre servicios
- AWS políticas gestionadas para Amazon QLDB
- Solución de problemas de identidad y acceso de Amazon QLDB

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que se realice en la QLDB.

Usuario de servicio: si utiliza el servicio de QLDB para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. Conforme vaya utilizando más características de QLDB para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en QLDB, consulte Solución de problemas de identidad y acceso de Amazon QLDB.

Administrador de servicio: si está a cargo de los recursos de QLDB en su empresa, probablemente tenga acceso completo a QLDB. Su trabajo consiste en determinar a qué características y recursos de QLDB deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con QLDB, consulte Cómo funciona Amazon QLDB con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a QLDB. Para consultar ejemplos de políticas basadas en la identidad de QLDB que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWSen la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede

asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para

obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

- Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations .
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en

función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon QLDB con IAM

Antes de utilizar IAM para administrar el acceso a QLDB, conozca qué características de IAM se pueden utilizar con QLDB.

Características de IAM que puede utilizar con Amazon QLDB

Característica de IAM	Compatibilidad de QLDB
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política	Sí
ACL	No
ABAC (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Permisos de entidades principales	No
Roles de servicio	Sí
Roles vinculados al servicio	No

Para obtener una visión general de cómo funcionan la QLDB y Servicios de AWS otras funciones con la mayoría de las funciones de IAM, Servicios de AWS consulte Cómo funcionan con IAM en la Guía del usuario de IAM.

Políticas basadas en identidades de QLDB

Compatibilidad con las políticas basadas en identidad	Sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está adjunto. Para más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades de QLDB

Para ver ejemplos de políticas basadas en identidad de QLDB, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Políticas basadas en recursos de QLDB

Compatibilidad con las políticas basadas en recursos	No

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico.

Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los directores pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para más información, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Acciones de política para QLDB

Admite acciones de política	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de las acciones de QLDB, consulte Acciones definidas por Amazon QLDB en la Referencia de autorizaciones de servicio.

Las acciones de políticas de QLDB utilizan el siguiente prefijo antes de la acción:

```
qldb
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "qldb:action1",
      "qldb:action2"
]
```

Puede utilizar caracteres comodín (*) para especificar varias acciones . Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "qldb:Describe*"
```

Para interactuar con la API de datos transaccionales de QLDB (sesión de QLDB) mediante la ejecución de instrucciones PartiQL en un libro mayor, debe conceder permiso a la acción `SendCommand` de la siguiente manera.

```
"Action": "qldb:SendCommand"
```

Para ver los libros mayores en el modo de permisos `STANDARD`, vaya a Referencia de permisos PartiQL y consulte los permisos adicionales necesarios para cada comando PartiQL.

Para ver ejemplos de políticas basadas en identidad de QLDB, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Recursos de políticas de QLDB

Admite recursos de políticas	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de tipos de recursos QLDB y sus ARN, consulte Recursos definidos por Amazon QLDB en la Referencia de autorizaciones de servicio. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por Amazon QLDB.

En QLDB, el recurso principal son los libros mayores. QLDB también admite otros tipos de recursos: tablas y secuencias. Sin embargo, puede crear tablas y secuencias solamente en el contexto de libro mayor existente.

Una tabQLDB es una vista materializada de una colección sin ordenar de revisiones de documentos del diario del libro mayor. En el modo de permisos `STANDARD` del libro mayor, deberá crear políticas de IAM que concedan permisos para ejecutar instrucciones PartiQL en este recurso de tabla. Con los permisos del recurso de tabla podrá ejecutar instrucciones para acceder al estado actual de la tabla. También puede consultar el historial de revisiones de la tabla usando la función integrada `history()`. Para obtener más información, consulte Introducción al modo de permisos estándar en Amazon QLDB.

> **Note**
>
> La `CREATE TABLE` instrucción crea una tabla con un identificador único y el nombre de tabla proporcionado. El nombre de tabla proporcionado debe ser único entre todas las tablas activas. Sin embargo, QLDB le permite desactivar tablas, por lo que puede haber varias tablas inactivas que compartan el mismo nombre de tabla. Por lo tanto, los ARN de los recursos de la tabla hacen referencia al identificador único asignado por el sistema y no al nombre de la tabla definido por el usuario.

Cada libro mayor también proporciona un recurso de catálogo definido por el sistema que puede consultar para obtener una lista de todas las tablas e índices de un libro mayor. Para obtener más información acerca del modelo de objeto de datos de QLDB, consulte Conceptos básicos y terminología de Amazon QLDB.

Estos recursos tienen ARN únicos asociados a ellos, tal y como se muestra en la siguiente tabla.

Tipo de recurso	ARN
ledger	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}`
table	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/table/${TableId}`
catalog	`arn:${Partition}:qldb:${Region}:${Account}:ledger/${LedgerName}/information_schema/user_tables`
stream	`arn:${Partition}:qldb:${Region}:${Account}:stream/${LedgerName}/${StreamId}`

Por ejemplo, para especificar el recurso `myExampleLedger` en su instrucción, utilice el siguiente ARN:

```
"Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
```

Para especificar varios recursos en una única instrucción, separe los ARN con comas.

```
"Resource": [
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger1",
      "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger2"
]
```

Para ver ejemplos de políticas basadas en identidad de QLDB, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Claves de condición de política para QLDB

Admite claves de condición de políticas específicas del servicio	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de las claves de condición de QLDB, consulte Claves de condición para Amazon QLDB en la Referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por Amazon QLDB.

Las acciones `PartiQLDropIndex` y `PartiQLDropTable` admiten la clave de condición `qldb:Purge`. Esta clave de condición filtra el acceso por el valor de `purge` especificado en una instrucción `DROP` PartiQL. Sin embargo, QLDB actualmente solo admite `purge = true` para instrucciones `DROP INDEX` y `purge = false` para instrucciones `DROP TABLE`. Otras acciones de QLDB admiten algunas claves de condición globales.

Para ver ejemplos de políticas basadas en identidad de QLDB, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Listas de control de acceso (ACL) en QLDB

Admite las ACL	No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Control de acceso basado en atributos (ABAC) con QLDB

Admite ABAC (etiquetas en las políticas)	Sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte ¿Qué es ABAC? en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Para obtener más información acerca del etiquetado de recursos de QLDB, consulte Etiquetado de recursos de Amazon QLDB.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Actualización de libros mayores de QLDB basada en etiquetas.

Uso de credenciales temporales con QLDB

Compatible con el uso de credenciales temporales	Sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluidas las que Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para más información sobre el cambio de roles, consulte Cambio a un rol (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para más información, consulte Credenciales de seguridad temporales en IAM.

Permisos de entidades principales entre servicios de QLDB

Admite sesiones de acceso directo (FAS)	No

Cuando utilizas un usuario o un rol de IAM para realizar acciones en AWSél, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar

ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Roles de servicio de QLDB

Compatible con roles de servicio	Sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de QLDB. Edite los roles de servicio solo cuando QLDB proporcione orientación para hacerlo.

QLDB admite roles de servicio para las operaciones de API `ExportJournalToS3` y `StreamJournalToKinesis`, tal como se describe en la siguiente sección.

Elección de un rol de IAM en QLDB

Cuando exporta bloques de diario de secuencia de QLDB, debe elegir un rol que permita a QLDB escribir objetos en el destino indicado en su nombre. Si ya ha creado un rol de servicio, QLDB le proporciona una lista de roles para elegir. Es importante elegir un rol que permita el acceso para escribir en el bucket de Amazon S3 especificado para una exportación o en el recurso de Amazon Kinesis Data Streams especificado para una secuencia. Para obtener más información, consulte Permisos de exportación de diarios en QLDB o Permisos de secuencia en QLDB.

> **Note**
>
> Para transferir un rol a QLDB al solicitar una secuencia o exportación de diario, debe tener permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM. Esto se suma a los permisos para realizar `qldb:ExportJournalToS3` en el recurso del libro mayor de QLDB o `qldb:StreamJournalToKinesis` en el subrecurso de secuencia de QLDB.

Roles vinculados a servicios de QLDB

Compatible con roles vinculados al servicio	No

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para obtener más información acerca de cómo crear o administrar roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Introducción al modo de permisos estándar en Amazon QLDB

Utilice esta sección para conocer el modo de permisos estándar en Amazon QLDB. En esta sección se proporciona una tabla de referencia que le ayudará a redactar una política basada en la identidad en AWS Identity and Access Management (IAM) para las acciones de PartiQL y los recursos de tablas en QLDB. También incluye un step-by-step tutorial para crear políticas de permisos en IAM e instrucciones para encontrar un ARN de tabla y crear etiquetas de tabla en QLDB.

Temas

- El modo de permisos STANDARD
- Referencia de permisos PartiQL
- Búsqueda de un ID y un ARN de tabla
- Etiquetar tablas
- Tutorial de inicio rápido: Creación de políticas de permisos

El modo de permisos **`STANDARD`**

QLDB ahora admite un modo de permisos `STANDARD` para los recursos del libro mayor. El modo de permisos estándar habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL. De forma predeterminada, este modo deniega todas las solicitudes para ejecutar cualquier comando de PartiQL en cualquier tabla de un libro mayor.

> **Note**
>
> Anteriormente, el único modo de permisos disponible para un libro mayor era `ALLOW_ALL`. El modo `ALLOW_ALL` habilita el control de acceso con granularidad de nivel de API para los libros mayores, y sigue siendo compatible, aunque no se recomienda, para los libros mayores de QLDB. Este modo permite a los usuarios que tengan el permiso de API `SendCommand` ejecutar todos los comandos de PartiQL en cualquier tabla del libro mayor que se especifique en la política de permisos (por lo tanto, los comandos «permitir todos» de PartiQL).
>
> Puede cambiar el modo de permisos de los libros mayores existentes de `ALLOW_ALL` a `STANDARD`. Para obtener más información, consulte Migrar al modo de permisos estándar.

Para permitir comandos en el modo estándar, debe crear una política de permisos en IAM para recursos de tabla y acciones PartiQL específicos. Esto se suma al permiso API `SendCommand` para el libro mayor. Para facilitar las políticas en este modo, QLDB introdujo un conjunto de acciones de IAM para los comandos PartiQL y nombres de recursos de Amazon (ARN) para las tablas de QLDB. Para obtener más información acerca del modelo de objeto de datos de QLDB, consulte Conceptos básicos y terminología de Amazon QLDB.

Referencia de permisos PartiQL

En la siguiente tabla se muestra cada comando PartiQL de QLDB, las acciones de IAM correspondientes para las que debe conceder permisos para ejecutar el comando AWS y los recursos para los que puede conceder los permisos. Las acciones se especifican en el campo `Action` de la política y el valor del recurso se especifica en el campo `Resource` de la política.

> **Important**
>
> - Las políticas de IAM que conceden permisos a estos comandos PartiQL solo se aplican a su libro mayor si el modo de permisos `STANDARD` está asignado al libro mayor. Estas políticas no se aplican a los libros mayores en el modo de permisos `ALLOW_ALL`.
>
> Para obtener información sobre cómo especificar el modo de permisos al crear o actualizar un libro mayor, consulte Operaciones básicas de libros mayores de Amazon QLDB o Paso 1: crear un nuevo libro mayor en la Introducción a la consola.

- Para ejecutar cualquier comando PartiQL en un libro mayor, también debe conceder permiso a la acción de la API `SendCommand` para el recurso del libro mayor. Esto se suma a las acciones de PartiQL y a los recursos de tabla que se muestran en la siguiente tabla. Para obtener más información, consulte Ejecutar transacciones de datos.

Comandos PartiQL de Amazon QLDB y permisos necesarios

Comando	Permisos necesarios (acciones de IAM)	Recursos	Acciones dependientes
CREATE TABLE	qldb:PartiQLCreateTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*	qldb:TagResource (para etiquetar en el momento de la creación)
DROP TABLE	qldb:PartiQLDropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
UNDROP TABLE	qldb:PartiQLUndropTable	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
CREATE INDEX	qldb:PartiQLCreateIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
DROP INDEX	qldb:PartiQLDropIndex	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	

Comando	Permisos necesarios (acciones de IAM)	Recursos	Acciones dependientes
DELETE FROM-REMOVE (para documentos completos)	qldb:PartiQLDelete	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
INSERT	qldb:PartiQLInsert	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	
UPDATE FROM (INSERT, REMOVE o SET)	qldb:PartiQLUpdate	arn:aws:qldb: *region*:*account-id* :ledger/*ledger-name* /table/*table-id*	qldb:PartiQLSelect
REDACT_REVISION (procedimiento almacenado)	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*	

Comando	Permisos necesarios (acciones de IAM)	Recursos	Acciones dependientes
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*	

Para ver ejemplos de documentos de políticas de IAM que conceden permisos a estos comandos PartiQL, vaya Tutorial de inicio rápido: Creación de políticas de permisos a o consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Búsqueda de un ID y un ARN de tabla

Para encontrar el ID de una tabla, utilice AWS Management Console o consulte la tabla information_schema.user_tables. Para ver los detalles de la tabla en la consola o consultar esta tabla del catálogo del sistema, debe tener permiso `SELECT` sobre el recurso del catálogo del sistema. Por ejemplo, para encontrar el ID de tabla de la tabla `Vehicle`, puede ejecutar la siguiente instrucción.

```
SELECT * FROM information_schema.user_tables
WHERE name = 'Vehicle'
```

Esta consulta devuelve resultados en un formato similar al del ejemplo siguiente.

```
{
    tableId: "Au1EiThbt8s0z9wM26REZN",
    name: "Vehicle",
```

```
    indexes: [
        { indexId: "Djg2nt0yIs2GY0T29Kud1z", expr: "[VIN]", status: "ONLINE" },
        { indexId: "4tPW3fUhaVhDinRgKRLhGU", expr: "[LicensePlateNumber]", status:
 "BUILDING" }
    ],
    status: "ACTIVE"
}
```

Para conceder permisos para ejecutar instrucciones PartiQL en una tabla, debe especificar un recurso de tabla en el siguiente formato ARN.

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

A continuación, se muestra un ejemplo de un ARN de tabla para el ID de tabla `Au1EiThbt8s0z9wM26REZN`.

```
arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/Au1EiThbt8s0z9wM26REZN
```

Mediante la consola

También puede utilizar la consola de QLDB para encontrar un ARN de tabla.

Cómo encontrar el ARN de una tabla (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Libros mayores.
3. En la lista de Libros mayores, elija el nombre del libro mayor cuyo ARN de tabla desee encontrar.
4. En la página de detalles del libro mayor, en la pestaña Tablas, busque el nombre de la tabla cuyo ARN desee encontrar. Para copiar el ARN, seleccione el icono de copia ( ) situado junto a él.

Etiquetar tablas

Puede etiquetar los recursos de tabla. Para administrar las etiquetas de las tablas existentes, utilice las AWS Management Console operaciones de la API `TagResourceUntagResource`,

y. `ListTagsForResource` Para obtener más información, consulte Etiquetado de recursos de Amazon QLDB.

> **Note**
>
> Los recursos de la tabla no heredan las etiquetas de su recurso de registro raíz. Actualmente, el etiquetado de tablas al crearlas solo se admite en los libros mayores en el modo de permisos `STANDARD`.

También puede definir etiquetas de tabla mientras crea la tabla mediante la consola de QLDB o especificándolas en una instrucción PartiQL `CREATE TABLE`. En el siguiente ejemplo, se crea una tabla llamada `Vehicle` con la etiqueta `environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Para etiquetar las tablas al crearlas, es necesario acceder a las acciones `qldb:PartiQLCreateTable` y `qldb:TagResource`.

Al etiquetar los recursos en el momento de su creación, ya no es necesario ejecutar scripts de etiquetado personalizados después de la creación del recurso. Una vez etiquetada una tabla, puede controlar el acceso a la tabla en función de esas etiquetas. Por ejemplo, puede conceder acceso total solo a las tablas que tengan una etiqueta específica. Para ver una política de ejemplo JSON, consulte Acceso completo a todas las acciones basadas en las etiquetas de las tablas.

Mediante la consola

También puede usar la consola de QLDB para definir las etiquetas de la tabla mientras crea la tabla.

Etiquetado de una tabla al crearla (consola)

1. Inicie sesión en la consola AWS Management Console de Amazon QLDB y ábrala en https://console.aws.amazon.com/qldb.
2. En el panel de navegación, elija Libros mayores.
3. En la lista de Libros mayores, elija el nombre del libro mayor donde quiere crear la tabla.
4. En la página de detalles del libro mayor, en la pestaña Tablas, elija Crear tabla.
5. En la página Crear tabla, haga lo siguiente:

- Nombre de tabla: introduzca un nombre para la tabla.
- Etiquetas: agregue metadatos a la tabla asociando etiquetas como pares de clave-valor. Puede agregar etiquetas a las tablas para que le resulte más fácil organizarlas e identificarlas.

  Elija Agregar etiqueta y, a continuación, introduzca cualquier par clave-valor según corresponda.

6. Cuando la configuración sea la deseada, elija Create table (Crear tabla).

Tutorial de inicio rápido: Creación de políticas de permisos

Este tutorial lo guía por los pasos para crear políticas de permisos en IAM para un libro mayor de Amazon QLDB en el modo de permisos `STANDARD`. A continuación, puede asignar los permisos a los usuarios, grupos o roles.

Para ver ejemplos de documentos de políticas de IAM que conceden permisos a estos comandos PartiQL y recursos de tabla, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

Temas

- Requisitos previos
- Creación de una política de solo lectura
- Crear una política de acceso completo
- Crear una política de solo lectura para una tabla específica
- Asignar permisos

Requisitos previos

Antes de comenzar, asegúrese de que hace lo siguiente:

1. Siga las instrucciones AWS de configuración que aparecen en el documentoAcceso a Amazon QLDB, si aún no lo ha hecho. Estos pasos incluyen registrarse AWS y crear un usuario administrativo.
2. Cree un nuevo libro mayor y elija el modo de permisos `STANDARD` para el libro mayor. Para aprender a hacerlo, consulte Paso 1: crear un nuevo libro mayor en Introducción a la consola o Operaciones básicas de libros mayores de Amazon QLDB.

Creación de una política de solo lectura

Para usar el editor de políticas JSON para crear una política de solo lectura para todas las tablas de un libro mayor en el modo de permisos estándar, haga lo siguiente:

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.
2. En la columna de navegación de la izquierda, elija Políticas.

   Si es la primera vez que elige Políticas, aparecerá la página Bienvenido a políticas administradas. Elija Comenzar.
3. En la parte superior de la página, seleccione Crear política.
4. Seleccione la pestaña JSON.
5. Copie y pegue la siguiente política JSON. Este ejemplo de política concede acceso de solo lectura a todas las tablas de un libro mayor.

   Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

```
{
  "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. Elija Revisar política.

> **Note**
>
> Puede alternar entre las pestañas Visual editor (Editor visual) y JSON en cualquier momento. Sin embargo, si realiza cambios o elige Review policy en la pestaña Visual editor, IAM podría reestructurar la política para optimizarla para el editor visual. Para obtener más información, consulte Reestructuración de política en la Guía del usuario de IAM.

7. En la página Review Policy (Revisar política), ingrese un Nombre y una descripción (opcional) para la política que está creando. Revise el Summary (Resumen) de la política para ver los permisos concedidos por su política. A continuación, elija Create policy (Crear política) para guardar su trabajo.

Crear una política de acceso completo

Para crear una política de acceso total para todas las tablas de un libro mayor de QLDB en el modo de permisos estándar, haga lo siguiente:

- Repita los pasos anteriores utilizando el siguiente documento de política. Esta política de ejemplo concede acceso a todos los comandos PartiQL de todas las tablas de un libro mayor, mediante el uso de caracteres comodín (*) para cubrir todas las acciones de PartiQL y todos los recursos de un libro mayor.

> **Warning**
>
> Este es un ejemplo del uso de un carácter comodín (*) para permitir todas las acciones de PartiQL, incluida la administración y las operaciones de lectura/escritura en todas las tablas de un libro mayor de QLDB. En su lugar, es una práctica recomendada especificar explícitamente cada acción que se va a conceder y solo lo que necesita ese usuario, rol o grupo.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Crear una política de solo lectura para una tabla específica

Para crear una política de acceso solo lectura para todas las tablas de un libro mayor de QLDB en el modo de permisos estándar, haga lo siguiente:

1. Busque el ARN de la tabla mediante AWS Management Console o consultando la tabla del catálogo del sistema. `information_schema.user_tables` Para ver instrucciones, consulte Búsqueda de un ID y un ARN de tabla.
2. Utilice el ARN de tabla para crear una política que permita el acceso de solo lectura a la tabla. Para ello, repita los pasos anteriores utilizando el siguiente documento de política.

Esta política de ejemplo concede acceso de solo lectura únicamente a la tabla especificada. En este ejemplo, el ID de tabla es `Au1EiThbt8s0z9wM26REZN`. Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedgery EiThbt Au1 8S0z9WM26ReZN* en el ejemplo por su propia información.

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

Asignar permisos

Después de crear una política de permisos de QLDB, asigne los permisos de la siguiente manera.

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en: AWS IAM Identity Center

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .
- Usuarios administrados en IAM a través de un proveedor de identidades:

Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades para Amazon QLDB

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de QLDB. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o la AWS API. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por QLDB, incluido el formato de los ARN para cada tipo de recurso, consulte Acciones, recursos y claves de condición para Amazon QLDB en la Referencia de autorizaciones de servicio.

Contenido

    - Acceso completo a todas las acciones
    - Acceso completo a todas las acciones basadas en las etiquetas de las tablas
    - Acceso de lectura/escritura
    - Acceso de solo lectura
    - Acceso de solo lectura a una tabla específica
    - Permitir el acceso para crear tablas
    - Permitir el acceso para crear tablas en función de las etiquetas de solicitud
- Exportación de un diario a un bucket de Amazon S3
- Transmisión de un diario a Kinesis Data Streams
- Actualización de libros mayores de QLDB basada en etiquetas

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de QLDB de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS

CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utilice el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Política de validación de Analizador de acceso de IAM en la Guía de usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para más información, consulte Configuración del acceso a una API protegido por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de QLDB

Para acceder a la consola de Amazon QLDB, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de QLDB en su. Cuenta de AWS Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles tengan acceso completo a la consola de QLDB y a todas sus funciones, adjunte la AWS siguiente política administrada a las entidades. Para obtener más información, consulte AWS políticas gestionadas para Amazon QLDB, y Adición de permisos para un usuario en la Guía del usuario de IAM.

```
AmazonQLDBConsoleFullAccess
```

Permisos de historial de consultas

Además de los permisos de QLDB, algunas características de la consola requieren permisos para el servicio de metadatos de consulta de bases de datos (prefijo de servicio: `dbqms`). Este es un servicio solo interno que administra sus consultas recientes y guardadas en el editor de consultas de la consola para QLDB y otros Servicios de AWS. Para obtener una lista completa de las acciones de la API del DBQMS, consulte el Servicio de metadatos de consultas de bases de datos en la Referencia de autorización del servicio.

Para permitir los permisos del historial de consultas, puede utilizar la política AWS gestionada AmazonQLDB. ConsoleFullAccess Esta política utiliza un comodín (`dbqms:*`) para permitir todas las acciones del DBQMS en todos los recursos.

También puede crear una política de IAM personalizada e incluir las siguientes acciones de DBQMS. El editor de consultas PartiQL de la consola de QLDB requiere permisos para usar estas acciones en las características del historial de consultas.

```
dbqms:CreateFavoriteQuery
dbqms:CreateQueryHistory
dbqms:DeleteFavoriteQueries
dbqms:DeleteQueryHistory
dbqms:DescribeFavoriteQueries
dbqms:DescribeQueryHistory
dbqms:UpdateFavoriteQuery
```

Permisos de acceso total a la consola sin historial de consultas

Para permitir el acceso total a la consola de QLDB sin ningún permiso de historial de consultas, puede crear una política de IAM personalizada que excluya todas las acciones del DBQMS. Por ejemplo, el siguiente documento de política permite los mismos permisos que concede la política AWS gestionada AmazonQLDB ConsoleFullAccess, excepto las acciones que comienzan con el prefijo de servicio. `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"
        }
      }
    }
  ]
}
```

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API o. AWS CLI AWS

```
{
    "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": "*"
        }
    ]
}
```

Ejecutar transacciones de datos

Para interactuar con la API de datos transaccionales de QLDB (sesión de QLDB) mediante la ejecución de instrucciones PartiQL en un libro mayor, debe conceder permiso a la acción `SendCommand` de la API. El siguiente documento JSON es un ejemplo de una política que concede permiso únicamente a la acción `SendCommand` de la API en el libro mayor `myExampleLedger`.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        }
    ]
}
```

Si `myExampleLedger` usa el modo de permisos `ALLOW_ALL`, esta política concede permisos para ejecutar todos los comandos PartiQL en cualquier tabla del libro mayor.

También puede utilizar una política AWS gestionada para conceder acceso total a todos los recursos de la QLDB. Para obtener más información, consulte AWS políticas gestionadas para Amazon QLDB.

Permisos estándar para acciones PartiQL y recursos de tabla

Para los libros mayores en el modo de permisos `STANDARD`, los siguientes documentos de política de IAM pueden servirle de ejemplo de casos de concesión de los permisos PartiQL adecuados. Para

obtener una lista de los permisos necesarios para cada comando de PartiQL, consulte la Referencia de permisos PartiQL.

Temas

- Acceso completo a todas las acciones
- Acceso completo a todas las acciones basadas en las etiquetas de las tablas
- Acceso de lectura/escritura
- Acceso de solo lectura
- Acceso de solo lectura a una tabla específica
- Permitir el acceso para crear tablas
- Permitir el acceso para crear tablas en función de las etiquetas de solicitud

Acceso completo a todas las acciones

El siguiente documento de política de JSON concede acceso completo para usar todos los comandos PartiQL en todas las tablas de `myExampleLedger`. Esta política produce el mismo efecto que el uso del modo de permisos `ALLOW_ALL` para el libro mayor.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Acceso completo a todas las acciones basadas en las etiquetas de las tablas

El siguiente documento de política de JSON utiliza un condición basada en las etiquetas de recurso de tabla para conceder acceso completo para usar todos los comandos PartiQL en todas las tablas de `myExampleLedger`. Los permisos se conceden solo si la etiqueta de la tabla `environment` tiene el valor `development`.

> **Warning**
>
> Este es un ejemplo del uso de un carácter comodín (*) para permitir todas las acciones de PartiQL, incluida la administración y las operaciones de lectura/escritura en todas las tablas de un libro mayor de QLDB. En su lugar, es una práctica recomendada especificar explícitamente cada acción que se va a conceder y solo lo que necesita ese usuario, rol o grupo.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Acceso de lectura/escritura

El siguiente documento de política de JSON otorga permisos para seleccionar, insertar, actualizar y eliminar datos en todas las tablas de `myExampleLedger`. Esta política no concede permisos para redactar datos ni modificar el esquema, por ejemplo, para crear y eliminar tablas e índices.

> **Note**
>
> Una instrucción UPDATE requiere permisos tanto para la acción `qldb:PartiQLUpdate` como para la `qldb:PartiQLSelect` de la tabla que se va a modificar. Cuando se ejecuta una instrucción UPDATE, esta realiza una operación de lectura además de la operación de actualización. Al requerir ambas acciones, se garantiza que solo se concederán permisos UPDATE a los usuarios que estén autorizados para leer el contenido de una tabla.
> Del mismo modo, una instrucción DELETE requiere permisos tanto para la acción `qldb:PartiQLDelete` como para la acción `qldb:PartiQLSelect`.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Acceso de solo lectura

El siguiente documento de política de JSON concede permisos de solo lectura en todas las tablas de `myExampleLedger`. Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Acceso de solo lectura a una tabla específica

El siguiente documento de política de JSON concede permisos de solo lectura en una tabla determinada de `myExampleLedger`. En este ejemplo, el ID de tabla es `Au1EiThbt8s0z9wM26REZN`.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedgery EiThbt Au1 8S0z9WM26ReZN* en el ejemplo por su propia información.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLReadOnlyPermissionsOnTable",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLSelect",
                "qldb:PartiQLHistoryFunction"
            ],
            "Resource": [
```

```
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/
table/Au1EiThbt8s0z9wM26REZN"
            ]
        }
    ]
}
```

Permitir el acceso para crear tablas

El siguiente documento de política de JSON concede permisos para crear tablas en `myExampleLedger`. La acción `qldb:PartiQLCreateTable` requiere permisos para el tipo de recurso de la tabla. Sin embargo, no se conoce el identificador de tabla de una tabla nueva en el momento en que se ejecuta una instrucción `CREATE TABLE`. Por lo tanto, una política que conceda el permiso `qldb:PartiQLCreateTable` debe usar un comodín (*) en la tabla ARN para especificar el recurso.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBSendCommandPermission",
            "Effect": "Allow",
            "Action": "qldb:SendCommand",
            "Resource": "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger"
        },
        {
            "Sid": "QLDBPartiQLCreateTablePermission",
            "Effect": "Allow",
            "Action": [
                "qldb:PartiQLCreateTable"
            ],
            "Resource": [
                "arn:aws:qldb:us-east-1:123456789012:ledger/myExampleLedger/table/*"
            ]
        }
    ]
}
```

Permitir el acceso para crear tablas en función de las etiquetas de solicitud

El siguiente documento de política de JSON utiliza una condición basada en la clave de contexto `aws:RequestTag` para conceder permiso para crear tablas en `myExampleLedger`. Los permisos se conceden solo si la etiqueta de la solicitud `environment` tiene el valor `development`. Para etiquetar las tablas al crearlas, es necesario acceder a las acciones `qldb:PartiQLCreateTable` y `qldb:TagResource`. Para obtener información sobre cómo etiquetar tablas al crearlas, consulteEtiquetar tablas.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Exportación de un diario a un bucket de Amazon S3

Paso 1: permisos de exportación de diarios QLDB

En el siguiente ejemplo, usted concede a un usuario sus Cuenta de AWS permisos para realizar la `qldb:ExportJournalToS3` acción en un recurso de registro de QLDB. También concede permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM que desea transferir al servicio QLDB. Esto es necesario para todas las solicitudes de exportación de diarios.

Para usar esta política, sustituya *us-east-1,* 123456789012 *y* qldb-s3-export en el *myExampleLedger*ejemplo por su propia información.

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

Paso 2: permisos en un bucket de Amazon S3

En el siguiente ejemplo, se utiliza un rol de IAM para conceder a QLDB acceso para escribir en uno de sus buckets de Amazon S3, `DOC-EXAMPLE-BUCKET`. Esto también es obligatorio para todas las exportaciones de diarios de QLDB.

Además del permiso `s3:PutObject`, la política también concede el permiso `s3:PutObjectAcl` que permite configurar los permisos de la lista de control de acceso (ACL) para un objeto.

Para utilizar esta regla de ejemplo, sustituya DOC-EXAMPLE-BUCKET en el ejemplo por el nombre del bucket de Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QLDBJournalExportS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

A continuación, asocie esta política de permisos a un rol de IAM que QLDB pueda asumir para acceder a su bucket de Amazon S3. El siguiente documento JSON es un ejemplo de una política de confianza que permite a QLDB asumir el rol de IAM para cualquier recurso de QLDB solo de la cuenta 123456789012.

Para usar esta política, sustituya *us-east-1* y *123456789012* del ejemplo por su propia información.

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

```
}
```

Transmisión de un diario a Kinesis Data Streams

Paso 1: permisos de secuencia del diario QLDB

En el siguiente ejemplo, usted concede a un usuario sus Cuenta de AWS permisos para realizar la `qldb:StreamJournalToKinesis` acción en todos los subrecursos de flujo de QLDB de un libro mayor. También concede permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM que desea transferir al servicio QLDB. Esto es necesario para todas las solicitudes de secuencias.

Para usar esta política, sustituya *us-east-1*, 123456789012 *qldb-kinesis-stream*y, en el ejemplo *myExampleLedger*, por su propia información.

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

Paso 2: permisos de Kinesis Data Streams

En el siguiente ejemplo, utiliza una función de IAM para conceder a QLDB acceso para escribir registros de datos en su transmisión de datos de Amazon Kinesis,. *stream-for-qldb* Esto también es obligatorio para todas las secuencias de diarios de QLDB.

Para usar esta política, sustituya *us-east-1*, 123456789012 *stream-for-qldb*y, en el ejemplo, por su propia información.

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

A continuación, asocie esta política de permisos a un rol de IAM que QLDB pueda asumir para acceder a su flujo de datos de Kinesis. El siguiente documento JSON es un ejemplo de una política de confianza que permite a QLDB asumir un rol de IAM para cualquier secuencia de QLDB solo de la cuenta 123456789012 del libro mayor `myExampleLedger`.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Actualización de libros mayores de QLDB basada en etiquetas

Puede utilizar las condiciones de su política basada en la identidad para controlar el acceso a los recursos de QLDB basados en etiquetas. En este ejemplo, se muestra cómo crear una política que permita actualizar un libro mayor. Sin embargo, los permisos solo se conceden si la etiqueta del libro mayor `Owner` tiene el valor del nombre de usuario de dicho usuario. Esta política también proporciona los permisos necesarios para llevar a cabo esta acción en la consola.

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

También puede adjuntar esta política al usuario de en su cuenta. Si un usuario llamado `richard-roe` intenta actualizar un libro mayor de QLDB, el libro mayor debe estar etiquetado como `Owner=richard-roe` o `owner=richard-roe`. De lo contrario, se le deniega el acceso. La clave

de la etiqueta de condición `0wner` coincide con los nombres de las claves de condición `0wner` y `owner` porque no distinguen entre mayúsculas y minúsculas. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

Prevención de la sustitución confusa entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre distintos servicios puede provocar el confuso problema de un diputado.

La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitar el confuso problema de los diputados, AWS proporciona herramientas que lo ayudan a proteger los datos de todos los servicios y los directores de servicio tienen acceso a los recursos de su cuenta.

Se recomienda utilizar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos para limitar los permisos que Amazon QLDB concede a otro servicio para el recurso. Si se utilizan ambas claves de contexto de condición global, el valor `aws:SourceAccount` y la cuenta del valor `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utilicen en la misma instrucción de política.

En la siguiente tabla se enumeran los valores posibles de `aws:SourceArn` para las operaciones de la API de QLDB `ExportJournalToS3` y `StreamsJournalToKinesis`. Estas operaciones entran dentro del ámbito de este problema de seguridad porque requieren que AWS Security Token Service (AWS STS) asuma una función de IAM que usted especifique.

Operación de la API	Servicio llamado	AWS: SourceArn
`ExportJournalToS3`	AWS STS (`AssumeRole`)	Permite que QLDB asuma el rol de cualquier recurso de QLDB de la cuenta: `arn:aws:qldb: us-east-1 :123456789012 :*`

Operación de la API	Servicio llamado	AWS: SourceArn
		Actualmente, QLDB solo admite este ARN comodín para la exportación de diarios.

`StreamsJournalToKinesis`	AWS STS (AssumeRole)	Permite que QLDB asuma el rol de una secuencia de QLDB específica:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /IiPT4brpZCqCq3f4MTHbYy`
		Nota: solo puede especificar un ID de secuencia en el ARN después de crear el recurso de secuencia. Con este ARN, puede permitir que el rol solo se use para una única secuencia de QLDB.
		Permite que QLDB asuma el rol de cualquier secuencia de QLDB de un libro mayor:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/myExample Ledger /*`
		Permite que QLDB asuma el rol de cualquier secuencia de QLDB de la cuenta:
		`arn:aws:qldb: us-east-1 :123456789012 :stream/*`
		Permite que QLDB asuma el rol de cualquier recurso de QLDB de la cuenta:
		`arn:aws:qldb: us-east-1 :123456789012 :*`

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodín (`*`) para las partes desconocidas del ARN, por ejemplo, `arn:aws:qldb:us-east-1:123456789012:*`.

El siguiente ejemplo de política de confianza para un rol de IAM muestra cómo se pueden utilizar las claves contextuales de condición global `aws:SourceArn` y `aws:SourceAccount` para evitar el problema del suplente confuso. Con esta política de confianza, QLDB puede asumir el rol de cualquier secuencia de QLDB en la cuenta `123456789012` del libro mayor `myExampleLedger` únicamente.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "qldb.amazonaws.com"
    },
    "Action": [ "sts:AssumeRole" ],
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:qldb:us-east-1:123456789012:stream/myExampleLedger/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

AWS políticas gestionadas para Amazon QLDB

Una política AWS administrada es una política independiente creada y administrada por. AWS AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Para obtener más información sobre las operaciones de la API de QLDB en AWS estas políticas administradas, consulte la. Referencia de la API de Amazon QLDB

Temas

- AWS política gestionada: AmazonQLDB ReadOnly
- AWS política gestionada: AmazonQLDB FullAccess
- AWS política gestionada: AmazonQLDB ConsoleFullAccess
- Actualizaciones de QLDB a las políticas gestionadas AWS

AWS política gestionada: AmazonQLDB ReadOnly

Utilice la ReadOnly política de AmazonQLDB para conceder permisos de solo lectura a todos los recursos de QLDB. Puede adjuntar esta política a sus identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos para el servicio `qldb`.

- Permite a las entidades principales describir y enumerar todos los recursos de QLDB y sus etiquetas. Estos recursos incluyen libros mayores, trabajos de exportación de Amazon S3 y secuencias a Kinesis Data Streams.
- Permite a las entidades principales obtener un bloque, un resumen o una revisión del diario de cualquier libro mayor para verificar los datos criptográficamente.
- No permite que las entidades principales ejecuten ningún comando de PartiQL en tablas de ningún libro mayor.

AWS política gestionada: AmazonQLDB FullAccess

Utilice la FullAccess política de AmazonQLDB para conceder permisos administrativos completos a todos los recursos de QLDB a través de la API de QLDB o la. AWS CLI Puede adjuntar esta política a sus identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `qldb`
  - Permite a las entidades principales crear, describir, enumerar y administrar todos los recursos de QLDB y sus etiquetas. Estos recursos incluyen libros mayores, trabajos de exportación de Amazon S3 y secuencias a Kinesis Data Streams.
  - Permite a las entidades principales ejecutar todos los comandos PartiQL en todas las tablas de cualquier libro mayor mediante el controlador de QLDB o el intérprete de comandos de QLDB.
  - Permite a las entidades principales obtener un bloque, un resumen o una revisión del diario de cualquier libro mayor para verificar los datos criptográficamente.
- `iam`: permite a las entidades principales transferir cualquier recurso de rol de IAM de su cuenta al servicio QLDB. Esto es necesario para todas las exportaciones de diarios y solicitudes de secuencias.

AWS política gestionada: AmazonQLDB ConsoleFullAccess

Utilice la ConsoleFullAccess política de AmazonQLDB para conceder permisos administrativos completos a todos los recursos de QLDB a través de la API de QLDB o AWS Management Console la. AWS CLI Puede adjuntar esta política a sus identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `qldb`
  - Permite a las entidades principales crear, describir, enumerar y administrar todos los recursos de QLDB y sus etiquetas. Estos recursos incluyen libros mayores, trabajos de exportación de Amazon S3 y secuencias a Kinesis Data Streams.

  - Permite a las entidades principales ejecutar todos los comandos PartiQL en todas las tablas de cualquier libro mayor mediante la consola de QLDB, el controlador de QLDB o el intérprete de comandos de QLDB.
  - Permite a las entidades principales insertar ejemplos de datos de aplicaciones en cualquier libro mayor mediante la consola de QLDB.
  - Permite a las entidades principales obtener un bloque, un resumen o una revisión del diario de cualquier libro mayor para verificar los datos criptográficamente.
- `dbqms`: permite a las entidades principales utilizar todas las acciones del Servicio de metadatos de consulta de bases de datos. Se trata de un servicio exclusivo interno que la consola de QLDB requiere para crear, describir y administrar las consultas recientes y guardadas para el editor de consultas PartiQL.
- `kinesis`: permite a las entidades principales describir y enumerar los recursos de Amazon Kinesis Data Streams. Estos recursos son los destinos objetivo en los que los recursos de secuencia de QLDB pueden escribir datos.
- `iam`: permite a las entidades principales transferir cualquier recurso de rol de IAM de su cuenta al servicio QLDB. Esto es necesario para todas las exportaciones de diarios y solicitudes de secuencias.

Actualizaciones de QLDB a las políticas gestionadas AWS

Consulte los detalles sobre las actualizaciones de las políticas AWS gestionadas para la QLDB desde que este servicio comenzó a realizar un seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página de historial de versiones de QLDB.

Cambio	Descripción	Fecha
AmazonQLDBFullAccess, AmazonQLDB: actualización de las políticas existentes ConsoleFullAccess	QLDB agregó un nuevo permiso para permitir a las entidades principales redactar las revisiones de los documentos en todos los	4 de noviembre de 2022

Cambio	Descripción	Fecha
	libros mayores en el modo de permisos `STANDARD`.	
AmazonQLDB, AmazonQLDB: actualización de las políticas existentes FullAccess ConsoleFullAccess	QLDB agregó nuevos permisos para que las entidades principales puedan transferir cualquier recurso de rol de IAM de su cuenta al servicio QLDB. Esto es necesario para todas las exportaciones de diarios y solicitudes de secuencias.	2 de septiembre de 2021
AmazonQLDB ReadOnly: actualización a una política existente	QLDB eliminó una acción `qldb:GetBlock` duplicada que anteriormente aparecía en la lista dos veces y reordenó el campo `"Effect"` para que aparezca antes que el campo `"Action"`.	1 de julio de 2021

Cambio	Descripción	Fecha
AmazonQLDB, AmazonQLDB: actualización de las FullAccess políticas existentes ConsoleFullAccess	QLDB agregó nuevos permisos para permitir a las entidades principales actualizar el modo de permisos en todos los libros mayores y ejecutar todos los comandos PartiQL en todos los libros mayores en el nuevo modo de permisos `STANDARD`. El modo de permisos `STANDARD` admite el control de acceso a nivel de tabla y la granularidad de los comandos PartiQL. Para facilitar el nuevo modo de permisos, QLDB introdujo un conjunto de acciones de IAM para los tipos de comandos PartiQL y nombres de recursos de Amazon (ARN) para los recursos de tabla de QLDB. Estas dos políticas se actualizan para incluir las nuevas acciones PartiQL que permiten el acceso completo a los libros mayores `STANDARD`.	27 de mayo de 2021
QLDB comenzó a realizar el seguimiento de los cambios	QLDB comenzó a realizar un seguimiento de los cambios en sus políticas gestionadas AWS .	1 de marzo de 2021

Solución de problemas de identidad y acceso de Amazon QLDB

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con QLDB e IAM.

Temas

No tengo autorización para realizar una acción en QLDB

Si AWS Management Console le indica que no está autorizado a realizar una acción, debe ponerse en contacto con su administrador para obtener ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

En el siguiente ejemplo, el error se produce cuando el usuario `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`myExampleLedger`*, pero no tiene los permisos ficticios `qldb:`*`DescribeLedger`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 qldb:DescribeLedger on resource: myExampleLedger
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *`myExampleLedger`* mediante la acción `qldb:`*`DescribeLedger`*.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a QLDB.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en QLDB. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Para obtener instrucciones sobre la solución de problemas relacionados con este error específico de las operaciones de exportación o secuencia del diario, consulte Solución de problemas de Amazon QLDB.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de QLDB

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para obtener información acerca de si QLDB admite estas características, consulte Cómo funciona Amazon QLDB con IAM.
- Para obtener información sobre cómo proporcionar acceso a sus recursos a través de los Cuentas de AWS que posee, consulte Proporcionar acceso a un usuario de IAM en otro de su propiedad en la Cuenta de AWS Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Registro y monitoreo en Amazon QLDB

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon QLDB y sus soluciones. AWS Debe recopilar datos de monitoreo de todas las partes de su AWS solución para poder depurar más fácilmente un error multipunto si se produce uno. No obstante, antes de comenzar a monitorizar QLDB, debe crear un plan de monitorización que incluya respuestas a las siguientes preguntas:

- ¿Cuáles son los objetivos de la supervisión?
- ¿Qué recursos va a supervisar?
- ¿Con qué frecuencia va a supervisar estos recursos?
- ¿Qué herramientas de monitoreo va a utilizar?
- ¿Quién se encargará de realizar las tareas de monitoreo?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso consiste en establecer un punto de referencia del desempeño de QLDB normal en su entorno. Para ello se mide el desempeño en distintos momentos y bajo distintas condiciones de carga. A medida que monitorice QLDB, almacene los datos de monitorización históricos para que pueda compararlos con los datos de rendimiento actual, identificar los patrones de rendimiento normal y las anomalías en el rendimiento, así como desarrollar métodos para la resolución de problemas.

Para establecer un punto de referencia debe, como mínimo, monitorizar los elementos siguientes:

- Lea y escriba las E/S y el almacenamiento para poder realizar un seguimiento de los patrones de consumo de su libro mayor a efectos de facturación.
- Controle la latencia, para que pueda realizar un seguimiento del rendimiento de su libro mayor al ejecutar operaciones de datos.
- Excepciones, para que pueda determinar si alguna solicitud ha provocado un error.

Temas

- Registrar llamadas a la API de Amazon QLDB con AWS CloudTrail

Herramientas de monitoreo

AWS proporciona varias herramientas que puede utilizar para supervisar Amazon QLDB. Puede configurar algunas de estas herramientas para que monitoricen por usted, pero otras herramientas requieren intervención manual. Le recomendamos que automatice las tareas de monitorización en la medida de lo posible.

Temas

- Herramientas de monitoreo automatizadas
- Herramientas de monitoreo manuales

Herramientas de monitoreo automatizadas

Puede utilizar las siguientes herramientas de monitorización automatizada para vigilar QLDB e informar cuando haya algún problema:

- Amazon CloudWatch Alarms: observe una sola métrica durante un período de tiempo que especifique y realice una o más acciones en función del valor de la métrica en relación con un umbral determinado durante varios períodos de tiempo. La acción es una notificación enviada a un tema del Servicio de Notificación Simple (Amazon SNS) o a una política de Auto Scaling de Amazon EC2. CloudWatch las alarmas no invocan acciones simplemente porque se encuentren en un estado determinado; el estado debe haber cambiado y se ha mantenido durante un número específico de períodos. Para obtener más información, consulte Monitorización con Amazon CloudWatch.
- Amazon CloudWatch Logs: supervise, almacene y acceda a sus archivos de registro desde AWS CloudTrail u otras fuentes. Para obtener más información, consulta Supervisión de archivos de registro en la Guía del CloudWatch usuario de Amazon.
- Amazon CloudWatch Events: haga coincidir los eventos y diríjalos a una o más funciones o transmisiones de destino para realizar cambios, capturar información de estado y tomar medidas correctivas. Para obtener más información, consulta Qué es Amazon CloudWatch Events en la Guía del CloudWatch usuario de Amazon.
- AWS CloudTrail Supervisión de registros: comparta archivos de registro entre cuentas, supervise los archivos de CloudTrail registro en tiempo real enviándolos a CloudWatch Logs, cree aplicaciones de procesamiento de registros en Java y valide que sus archivos de registro no hayan

cambiado después de su entrega CloudTrail. Para obtener más información, consulte Trabajar con archivos de CloudTrail registro en la Guía del AWS CloudTrail usuario.

Herramientas de monitoreo manuales

Otra parte importante de la supervisión de la QLDB implica la supervisión manual de los elementos que CloudWatch las alarmas no cubren. La QLDB CloudWatch, Trusted Advisor, y AWS Management Console otros paneles proporcionan at-a-glance una vista del estado de su entorno. AWS Le recomendamos que también verifique los archivos de registro en Amazon QLDB.

- El panel de QLDB muestra lo siguiente:
  - E/S de lectura y escritura
  - Almacenamiento indexado y del diario
  - Latencia de comandos
  - Excepciones
- La página de CloudWatch inicio muestra lo siguiente:
  - Alarmas y estado actual
  - Gráficos de alarmas y recursos
  - Estado de los servicios

  Además, se puede utilizar CloudWatch para hacer lo siguiente:
  - Crear paneles personalizados para monitorizar los servicios que le interesan
  - Realizar un gráfico con los datos de las métricas para resolver problemas y descubrir tendencias
  - Busque y explore todas sus métricas AWS de recursos
  - Crear y editar las alarmas de notificación de problemas

Monitorización con Amazon CloudWatch

Puede monitorizar Amazon QLDB CloudWatch mediante métricas legibles, que recopila y procesa datos sin procesar de Amazon QLDB. near-real-time Estas estadísticas se mantienen durante dos semanas, de forma que pueda acceder a información histórica y disponer de una mejor perspectiva sobre el desempeño de su aplicación web o servicio. De forma predeterminada, los datos de las métricas de QLDB se envían CloudWatch automáticamente en períodos de 1 o 15 minutos. Para obtener más información, consulta ¿Qué son Amazon CloudWatch, Amazon CloudWatch Events y Amazon CloudWatch Logs? en la Guía del CloudWatch usuario de Amazon.

Temas

¿Cómo utilizo las métricas de QLDB?

Las métricas mostradas por QLDB proporcionan información que puede analizar de diferentes maneras. En la siguiente lista se indican algunos usos frecuentes de las métricas. Se trata de sugerencias que puede usar como punto de partida y no de una lista completa.

- Puede monitorizar `JournalStorage` y `IndexedStorage` durante un período de tiempo específico para realizar un seguimiento del espacio en disco que consume su libro mayor.
- Puede monitorizar `ReadIOs` y `WriteIOs` durante un período de tiempo específico para realizar un seguimiento del número de solicitudes que procesa su libro mayor.
- Puede monitorizar `CommandLatency` para realizar un seguimiento del rendimiento del libro mayor en lo que respecta a las operaciones de datos y analizar los tipos de comandos que generan la mayor latencia.

Métricas y dimensiones de Amazon QLDB

Cuando interactúa con Amazon QLDB, envía las siguientes métricas y dimensiones a. CloudWatch Las métricas de almacenamiento se publican cada 15 minutos, y todas las demás métricas se agregan y se publican cada minuto. Puede utilizar los siguientes procedimientos para consultar las métricas de QLDB.

Para ver las métricas mediante la consola CloudWatch

Las métricas se agrupan en primer lugar por el espacio de nombres de servicio y, a continuación, por las diversas combinaciones de dimensiones dentro de cada espacio de nombres.

1. Abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Si es necesario, cambie la región. En la barra de navegación, seleccione la región donde residen sus recursos de AWS . Para obtener más información, consulte Regiones y puntos de enlace.
3. En el panel de navegación, seleccione Métricas.
4. En la pestaña Todas las métricas, elija QLDB.

Para ver las métricas mediante el AWS CLI

- En el símbolo del sistema, ejecute el siguiente comando.

```
aws cloudwatch list-metrics --namespace "AWS/QLDB"
```

CloudWatch muestra las siguientes métricas para la QLDB.

Dimensiones y métricas de Amazon QLDB

Aquí se muestran las métricas y dimensiones que Amazon QLDB envía a CloudWatch Amazon.

Métricas de QLDB

Métrica	Descripción
`JournalStorage`	La cantidad total de espacio en disco utilizada por el diario del libro mayor, publicada en intervalos de 15 minutos. El diario contiene el historial completo, inmutable y verificable de todos los cambios realizados en los datos. Unidades: `Bytes` Dimensiones: `LedgerName`
`IndexedStorage`	La cantidad total de espacio en disco que utilizan las tablas, los índices y el historial indexado del libro mayor, publicada en intervalos de 15 minutos. El almacenamiento indexado consiste en datos de libro mayor optimizados para consultas de alto rendimiento. Unidades: `Bytes` Dimensiones: `LedgerName`
`ReadIOs`	Es el número de solicitudes de E/S de lectura publicada s en intervalos de un minuto. Esto captura todos los tipos de operaciones de lectura, incluidas las transacciones de

Métrica	Descripción
	datos, las solicitudes de verificación, las exportaciones de diarios y las secuencias de diarios. Unidades: `Count` Dimensiones: `LedgerName`
`WriteIOs`	Es el número de solicitudes de E/S de escritura publicadas en intervalos de un minuto. Unidades: `Count` Dimensiones: `LedgerName`
`CommandLatency`	El tiempo necesario para las operaciones de datos, publicado en intervalos de un minuto. Unidades: `Milliseconds` Dimensiones: `CommandType, LedgerName`
`IsImpaired`	El indicador que señala si la secuencia de un diario a Kinesis Data Streams está dañada y se publica en intervalos de un minuto. Un valor de 1 indica que la secuencia está dañada y `0` indica lo contrario. Unidades: `Boolean` (0 o 1) Dimensiones: `LedgerName, StreamId`
`OccConflictExceptions`	El número de solicitudes a QLDB que generan una `OccConflictException` . Para obtener información sobre el control de concurrencia optimista (OCC), consulte Modelo de concurrencia de Amazon QLDB. Unidades: `Count`

Métrica	Descripción
`Session4xxExceptions`	El número de solicitudes a QLDB que generan un error HTTP 4xx. Unidades: `Count`
`Session5xxExceptions`	El número de solicitudes a QLDB que generan un error HTTP 5xx. Unidades: `Count`
`SessionRateExceededExceptions`	El número de solicitudes a QLDB que generan una `SessionRateExceededException` . Unidades: `Count`

Dimensiones de las métricas de QLDB

Las métricas de QLDB se califican según los valores para la cuenta, el nombre del libro mayor, el ID de la secuencia o el tipo de comando. Puede utilizar la CloudWatch consola para recuperar datos de QLDB en cualquiera de las dimensiones de la siguiente tabla.

Dimensión	Descripción
`LedgerName`	Esta dimensión limita los datos a un libro mayor específico. Este valor puede ser cualquier nombre de registro de la versión actual Región de AWS y la actual. Cuenta de AWS
`StreamId`	Esta dimensión limita los datos a una secuencia de diario específica. Este valor puede ser cualquier identificador de flujo de un libro mayor en el actual Región de AWS y en el actual. Cuenta de AWS
`CommandType`	Esta dimensión limita los datos a uno de los siguientes comandos de la API de datos de QLDB: • `AbortTransaction` • `CommitTransaction`

Dimensión	Descripción
	• `EndSession` • `ExecuteStatement` • `FetchPage` • `StartSession` • `StartTransaction` Para obtener información sobre cómo QLDB utiliza estos comandos para administrar las operaciones de datos, consulte Gestión de sesiones con el controlador.

Creación de CloudWatch alarmas para monitorear Amazon QLDB

Puedes crear una CloudWatch alarma de Amazon que envíe un mensaje del Servicio de Notificación Simple de Amazon (Amazon SNS) cuando la alarma cambie de estado. Una alarma vigila una métrica determinada durante el periodo especificado. Realiza una o varias acciones según el valor de la métrica con respecto a un umbral dado durante varios períodos de tiempo. La acción es una notificación que se envía a un tema de Amazon SNS o a una política de escalado automático.

Las alarmas invocan acciones únicamente en caso de cambios de estado sostenidos. CloudWatch las alarmas no invocan acciones simplemente porque se encuentran en un estado determinado. El estado debe haber cambiado y debe mantenerse durante el número de periodos especificado.

Para obtener más información sobre la creación de CloudWatch alarmas, consulta Uso de CloudWatch alarmas de Amazon en la Guía del CloudWatch usuario de Amazon.

Automatización de Amazon QLDB con eventos CloudWatch

Amazon CloudWatch Events le permite automatizar Servicios de AWS y responder automáticamente a los eventos del sistema, como los problemas de disponibilidad de las aplicaciones o los cambios de recursos. Los eventos de Servicios de AWS se envían a CloudWatch Events prácticamente en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas. Entre las acciones que se pueden activar automáticamente se incluyen las siguientes:

- Invocar una función AWS Lambda

- Invocar Ejecutar comando de Amazon EC2
- Desviar el evento a Amazon Kinesis Data Streams
- Activar una máquina de AWS Step Functions estados
- Notificar un tema de Amazon SNS o una cola de Amazon SQS

Amazon QLDB informa de un evento CloudWatch a Events cada vez que cambia el estado de un recurso contable en usted. Cuenta de AWS Actualmente, los eventos se emiten de forma garantizada, at-least-once únicamente para los recursos del libro mayor de la QLDB.

A continuación se muestra un ejemplo de un evento publicado por QLDB, en el que el estado de un libro mayor cambió a `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"
    }
}
```

Algunos ejemplos del uso de CloudWatch eventos con QLDB pueden incluir, entre otros, los siguientes:

- Activar una función de Lambda siempre que se cree inicialmente un nuevo libro mayor con estado `CREATING` y, finalmente, se convierte en `ACTIVE`.
- Notificar a un tema de Amazon SNS cuando el estado de su libro mayor cambia a `DELETING` y luego a `DELETED`.

Para obtener más información, consulta la Guía del usuario de Amazon CloudWatch Events.

Registrar llamadas a la API de Amazon QLDB con AWS CloudTrail

Amazon QLDB está integrado AWS CloudTrail con un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un miembro de la QLDB. Servicio de AWS CloudTrail captura todas las llamadas a la API de administración de recursos para la QLDB como eventos. Las llamadas capturadas incluyen las realizadas desde la consola de QLDB y las llamadas de código a las operaciones de la API de QLDB. Si crea un registro, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon Simple Storage Service (Amazon S3), incluidos los eventos de QLDB. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a la QLDB, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, incluido cómo configurarlo y habilitarlo, consulte la Guía del AWS CloudTrail usuario.

Información sobre QLDB en CloudTrail

CloudTrail está habilitada en su cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad compatible en la QLDB, esa actividad se registra en CloudTrail un evento junto con Servicio de AWS otros eventos en el historial de eventos. Puede ver, buscar y descargar eventos recientes en su. Cuenta de AWS Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos en su territorio Cuenta de AWS, incluidos los eventos de la QLDB, cree un sendero. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros Servicios de AWS para que analicen más a fondo los datos de eventos recopilados en los CloudTrail registros y actúen en función de ellos.

Para obtener más información, consulte los siguientes temas en la Guía del usuario de AWS CloudTrail :

- Introducción a la creación de registros de seguimiento
- CloudTrail servicios e integraciones compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail

- Recibir archivos de CloudTrail registro de varias regiones
- Recibir archivos de CloudTrail registro de varias cuentas

Todas las acciones de la API de gestión de recursos y datos no transaccionales de QLDB se registran CloudTrail y se documentan en la referencia de la API de Amazon QLDB. Por ejemplo, las llamadas a las acciones y a las `CreateLedger` `DeleteLedger` acciones generan `DescribeLedger` entradas en los archivos de registro. CloudTrail

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- si la solicitud se realizó con las credenciales del nodo raíz o del usuario
- si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado
- Si la solicitud la realizó otra persona Servicio de AWS

Para obtener más información, consulte el elemento CloudTrail UserIdentity.

Comprender las entradas de archivos de registro de QLDB

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que muestra estas acciones:

- `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"
    }
  ]
}
```

Validación de la conformidad para Amazon QLDB

Los auditores externos evalúan la seguridad y el cumplimiento de Amazon QLDB como parte de AWS varios programas de cumplimiento, incluidos, entre otros, los siguientes:

- Controles del Sistema y Organizaciones (System and Organization Controls, SOC)
- Industria de tarjetas de pago (PCI)
- Organización Internacional de Normalización (ISO)
- Programa de evaluación y administración de seguridad de sistemas de información (ISMAP)
- Ley de Portabilidad y Responsabilidad de Seguros Médicos de EE. UU (Health Insurance Portability and Accountability Act, HIPAA).

> **Note**
>
> Esta no es la lista completa de certificaciones QLDB de Amazon.

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de conformidad específicos, consulte Servicios de AWS Alcance por programa de conformidad Servicios de AWS y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Guías de inicio rápido sobre seguridad y cumplimiento: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- Diseño de arquitectura para garantizar la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: en este documento técnico se describe cómo pueden utilizar AWS las empresas para crear aplicaciones aptas para la HIPAA.

> **Note**
>
> No Servicios de AWS todas cumplen con los requisitos de la HIPAA. Para más información, consulte la Referencia de servicios compatibles con HIPAA.

- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en Amazon QLDB

La infraestructura AWS global se basa Regiones de AWS en zonas de disponibilidad. Regiones de AWS proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad Regiones de AWS y las zonas de disponibilidad, consulte Infraestructura global.AWS

Durabilidad del almacenamiento

El almacenamiento de diarios de QLDB presenta una replicación sincrónica en múltiples zonas de disponibilidad en las confirmaciones de transacciones. Esto garantiza que incluso un error total en el almacenamiento del diario en la zona de disponibilidad no comprometa la integridad de los datos ni la capacidad de mantener un servicio activo. Además, el diario de QLDB incluye archivos asíncronos para un almacenamiento tolerante a errores. Esta característica permite la recuperación de desastres en el caso, muy poco probable, de que se produzca un fallo de almacenamiento simultáneo en varias zonas de disponibilidad.

El almacenamiento indexado de QLDB está respaldado por la replicación en múltiples zonas de disponibilidad. Esto garantiza que incluso un error total en el almacenamiento indexado en la zona de disponibilidad no comprometa la integridad de los datos ni la capacidad de mantener un servicio activo.

Características de durabilidad de los datos

Además de la infraestructura AWS global, la QLDB ofrece las siguientes funciones para ayudar a respaldar sus necesidades de respaldo y resiliencia de datos.

Características del servicio QLDB

Exportación de diarios bajo demanda

QLDB proporciona una característica de exportación de diarios bajo demanda. Acceda al contenido de su diario exportando bloques de su libro mayor a un bucket de Amazon S3. Puede

utilizar estos datos para diversos fines, como la retención de datos, el análisis y la auditoría. Para obtener más información, consulte Exportación de datos de diarios desde Amazon QLDB.

Copia de seguridad y restauración

Por el momento, no se admite la restauración automática para las exportaciones. La exportación proporciona una capacidad básica para crear redundancia de datos adicional con la frecuencia definida. Sin embargo, un escenario de restauración depende de la aplicación, en el que los registros exportados presumiblemente se vuelven a escribir en un nuevo libro mayor utilizando el mismo método de ingesta o uno similar.

En este momento, QLDB no proporciona una característica específica de copia de seguridad y restauración relacionada.

Secuencias de diario

QLDB también proporciona una capacidad de secuencia continua de diarios. Puede integrar las secuencias de diarios de QLDB con la plataforma de streaming Amazon Kinesis para procesar los datos de los diarios en tiempo real. Para obtener más información, consulte Transmisión de datos de diarios desde Amazon QLDB.

Característica de diseño de QLDB

QLDB está diseñada para ser resistente a los daños lógicos. El diario QLDB es inmutable, lo que garantiza que todas las transacciones confirmadas se conserven en el diario. Además, se registran todos los cambios en los documentos confirmados, ya que esto permite ver cualquier cambio imprevisto en los datos del libro point-in-timemayor.

QLDB no proporciona una característica de recuperación automática para escenarios de daños lógicos en este momento.

Seguridad de la infraestructura en Amazon QLDB

Como servicio gestionado, Amazon QLDB está protegido por los procedimientos de seguridad de red global que se describen en AWS el documento técnico Amazon Web Services: Overview of Security Processes.

Utiliza las llamadas a la API AWS publicadas para acceder a la QLDB a través de la red. Los clientes deben ser compatibles con la seguridad de la capa de transporte (TLS) 1.0 o una versión posterior. Recomendamos TLS 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman

(DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante credenciales programáticas que estén asociadas a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

También puede usar un punto de conexión de una nube privada virtual (VPC) de QLDB. Los puntos de conexión de VPC permiten que los recursos de Amazon VPC utilicen sus direcciones IP privadas para acceder a QLDB sin exponerse en la Internet pública. Para obtener más información, consulte Acceso a Amazon QLDB mediante un punto de conexión de interfaz (AWS PrivateLink).

Acceso a Amazon QLDB mediante un punto de conexión de interfaz (AWS PrivateLink)

Puede utilizarla AWS PrivateLink para crear una conexión privada entre su VPC y Amazon QLDB. Puede acceder a la QLDB como si estuviera en su VPC, sin utilizar una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. AWS Direct Connect Las instancias de la VPC no necesitan direcciones IP públicas para acceder a QLDB.

Esta conexión privada se establece mediante la creación de un punto de conexión de interfaz alimentado por AWS PrivateLink. Creamos una interfaz de red de punto de conexión en cada subred habilitada para el punto de conexión de interfaz. Se trata de interfaces de red administradas por el solicitante que sirven como punto de entrada para el tráfico destinado a QLDB.

Para obtener más información, consulte Acceso a los Servicios de AWS a través de AWS PrivateLink en la Guía de AWS PrivateLink .

Temas

- Consideraciones sobre QLDB
- Creación de un punto de conexión de interfaz para QLDB
- Creación de una política de puntos de conexión para el punto de conexión de interfaz
- Disponibilidad de puntos de conexión de interfaz para QLDB

Consideraciones sobre QLDB

Antes de configurar un punto de conexión de interfaz para QLDB, consulte Consideraciones en la Guía de AWS PrivateLink .

> **Note**
>
> QLDB solo admite realizar llamadas a la API de datos transaccionales de la sesión de QLDB a través del punto de conexión de la interfaz. Esta API incluye solo la operación. SendCommand En el modo de permisos `STANDARD` de un libro mayor, puede controlar los permisos para acciones PartiQL específicas en esta API.

Creación de un punto de conexión de interfaz para QLDB

Puede crear un punto final de interfaz para la QLDB mediante la consola Amazon VPC o el (). AWS Command Line Interface AWS CLI Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía de AWS PrivateLink .

Cree un punto de conexión de interfaz para QLDB con el siguiente nombre de servicio:

```
com.amazonaws.region.qldb.session
```

Si habilita DNS privado para el punto de conexión de interfaz, puede realizar solicitudes API a QLDB usando su nombre de DNS predeterminado para la región. Por ejemplo, `session.qldb.us-east-1.amazonaws.com`.

Creación de una política de puntos de conexión para el punto de conexión de interfaz

Una política de punto de conexión es un recurso de IAM que puede adjuntar al punto de conexión de su interfaz. La política de puntos de conexión predeterminada permite acceso completo a QLDB a través del punto de conexión de interfaz. Para controlar el acceso permitido a QLDB desde la VPC, adjunte una política de puntos de conexión personalizada al punto de conexión de interfaz.

Una política de punto de conexión especifica la siguiente información:

- Las entidades principales que pueden llevar a cabo acciones (Cuentas de AWS, usuarios y roles).
- Las acciones que se pueden realizar.
- El recurso en el que se pueden realizar las acciones.

Para obtener más información, consulte Control del acceso a los servicios con políticas de punto de conexión en la Guía del usuario de AWS PrivateLink .

También puede usar el campo `Condition` en una política asociada a un usuario, grupo o rol para permitir el acceso solo desde un punto de conexión de interfaz específico. Si se usan juntas, las políticas de punto de conexión y las políticas de IAM pueden restringir el acceso a acciones específicas de QLDB en libros mayores específicos a un punto de conexión de interfaz específico.

Ejemplo de política de punto de conexión: restringir el acceso a un libro mayor de QLDB específico

A continuación, se muestra un ejemplo de una política de un punto de conexión personalizado para QLDB. Cuando se adjunta esta política a un punto de conexión de interfaz, se concede acceso a las acciones de `SendCommand` y a las acciones de solo lectura de PartiQL para todas las entidades principales en el recurso de libro mayor específico. En este ejemplo, el libro mayor debe estar en el modo de permisos `STANDARD`.

Para usar esta política, sustituya *us-east-1*, 123456789012 *myExampleLedger*y, en el ejemplo, por su propia información.

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

Ejemplo de política de IAM: restringir el acceso a un libro mayor de QLDB únicamente desde un punto de conexión de interfaz específico

A continuación se muestra un ejemplo de una política de IAM basada en identidades para QLDB. Al asociar esta política a un usuario, rol o grupo, se permite a `SendCommand` el acceso a un recurso del libro mayor solo desde el punto de conexión de interfaz especificado.

Para usar esta política, sustituya *us-east-1,* 123456789012 *y* vpce-1a2b3c4d en el *myExampleLedger*ejemplo por su propia información.

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

Disponibilidad de puntos de conexión de interfaz para QLDB

Amazon QLDB admite puntos de conexión de interfaz con políticas en todas las Regiones de AWS donde esté disponible QLDB. Para ver una lista completa de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS.

Solución de problemas de Amazon QLDB

Las siguientes secciones ofrecen la lista de los errores más comunes que puede encontrar al usar Amazon QLDB así como una guía sobre cómo solucionarlos.

Para obtener una guía de solución de problemas específica para el acceso a IAM, consulte Solución de problemas de identidad y acceso de Amazon QLDB.

Para conocer las mejores prácticas para ajustar las instrucciones PartiQL, consulte Optimizar el rendimiento de las consultas.

Temas

Ejecución de transacciones mediante el controlador QLDB

En esta sección se enumeran las excepciones más comunes que el controlador QLDB de Amazon puede devolver cuando lo utiliza para ejecutar transacciones PartiQL en un libro mayor. Para obtener más información acerca de esta característica, consulte Introducción al controlador. Para obtener información sobre las prácticas recomendadas para configurar y usar el controlador, consulte Recomendaciones de controladores.

Cada excepción incluye el mensaje de error específico, seguido de una breve descripción y sugerencias de posibles soluciones.

CapacityExceededException

Mensaje: Capacidad superada

Amazon QLDB rechazó la solicitud porque superaba la capacidad de procesamiento del libro mayor. La QLDB impone un límite de escalado interno por libro mayor para mantener el estado y el rendimiento del servicio. Este límite varía según el tamaño de la carga de trabajo de cada solicitud individual. Por ejemplo, una solicitud puede tener una mayor carga de trabajo si realiza

transacciones de datos ineficientes, como los escaneos de tablas que resultan de una consulta no apta para indexar.

Le recomendamos que espere antes de volver a intentar la solicitud. Si su solicitud encuentra esta excepción de forma constante, optimice sus instrucciones y reduzca la frecuencia y el volumen de las solicitudes que envía al libro mayor. Algunos ejemplos de optimización de instrucciones incluyen ejecutar menos instrucciones por transacción y ajustar los índices de las tablas. Para obtener información sobre cómo optimizar las instrucciones y evitar el escaneo de tablas, consulte Optimizar el rendimiento de las consultas.

Siempre recomendamos usar la versión más reciente del controlador QLDB. El controlador tiene una política de reintentos predeterminada que utiliza el Retroceso exponencial y la fluctuación de fase para volver a intentarlo automáticamente en excepciones como esta. El concepto de retroceso exponencial se basa en utilizar tiempos de espera progresivamente más largos entre reintentos para las respuestas a errores consecutivos.

InvalidSessionException

Mensaje: Transaction *`transactionId`* has expired

Una transacción ha superado su vida útil máxima. Una transacción puede ejecutarse durante un máximo de 30 segundos antes de confirmarse. Tras este límite de tiempo de espera, se rechaza cualquier trabajo realizado en la transacción y la QLDB descarta la sesión. Este límite evita que el cliente pierda sesiones al iniciar transacciones y no confirmarlas ni cancelarlas.

Si se trata de una excepción habitual en su aplicación, es probable que las transacciones simplemente estén tardando demasiado en ejecutarse. Si el tiempo de ejecución de la transacción supera los 30 segundos, optimice sus instrucciones para acelerar las transacciones. Algunos ejemplos de optimización de instrucciones incluyen ejecutar menos instrucciones por transacción y ajustar los índices de las tablas. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

InvalidSessionException

Mensaje: Session *`sessionId`* has expired

La QLDB descartó la sesión porque excedió su vida útil total máxima. La QLDB descarta las sesiones después de 13 a 17 minutos, independientemente de si hay una transacción activa. Las sesiones se pueden perder o dañar por distintos motivos, como fallos de hardware, fallos de red o reinicio de las aplicaciones. QLDB impone una duración máxima a las sesiones para garantizar que el software cliente sea resiliente a fallos de sesión.

Si se produce esta excepción, le recomendamos que inicie una sesión nueva y vuelva a intentar la transacción. También recomendamos utilizar la última versión del controlador QLDB, que administra el grupo de sesiones y su estado en nombre de la aplicación.

InvalidSessionException

Mensaje: No such session

El cliente intentó realizar transacciones con QLDB mediante una sesión que no existe. Suponiendo que el cliente esté utilizando una sesión que existía anteriormente, es posible que la sesión ya no exista debido a uno de los siguientes motivos:

- Si una sesión está involucrada en un fallo interno del servidor (es decir, un error con el código de respuesta HTTP 500), QLDB podría optar por descartar la sesión por completo, en lugar de permitir que el cliente realice transacciones con una sesión de estado incierto. En ese caso, cualquier intento de reintento en esa sesión fallará y se generará este error.
- QLDB finalmente olvida las sesiones caducadas. De ahí que, cualquier intento de continuar usando la sesión produzca este error, en lugar del `InvalidSessionException` inicial.

Si se produce esta excepción, le recomendamos que inicie una sesión nueva y vuelva a intentar la transacción. También recomendamos utilizar la última versión del controlador QLDB, que administra el grupo de sesiones y su estado en nombre de la aplicación.

RateExceededException

Mensaje: The rate was exceeded

QLDB limitó a un cliente en función de la identidad de la persona que llamaba. QLDB impone la limitación por región y por cuenta mediante un algoritmo de limitación de bucket de token. Lo hace para mejorar el rendimiento del servicio y garantizar un utilización justa para todos los clientes de QLDB. Por ejemplo, intentar adquirir una gran cantidad de sesiones simultáneas mediante la operación `StartSessionRequest` podría provocar una limitación.

Para mantener el buen estado de la aplicación y reducir aún más las limitaciones, puede volver a intentarlo con esta excepción mediante el Retroceso exponencial y la fluctuación de fase. El concepto de retroceso exponencial se basa en utilizar tiempos de espera progresivamente más largos entre reintentos para las respuestas a errores consecutivos. Siempre recomendamos usar la versión más reciente del controlador QLDB. El controlador tiene una política de reintentos predeterminada que utiliza el Retroceso exponencial y la fluctuación de fase para volver a intentarlo automáticamente en excepciones como esta.

La última versión del controlador QLDB también puede ayudar si QLDB limita constantemente su aplicación para llamadas `StartSessionRequest`. El controlador mantiene un conjunto de sesiones que se reutilizan en todas las transacciones, lo que puede ayudar a reducir el número de llamadas `StartSessionRequest` que realiza su aplicación. Para solicitar un aumento de los límites de la limitación API, contacte con el Centro AWS Support.

LimitExceededException

Mensaje: Exceeded the session limit

Un libro mayor ha superado su cuota (también conocida como límite) en cuanto al número de sesiones activas. Esta cuota se define en Cuotas y límites de Amazon QLDB. Con el tiempo, el recuento de sesiones activas de un libro mayor es uniforme, y los libros que se acerquen constantemente a la cuota podrían ver esta excepción periódicamente.

Para mantener el buen estado de la aplicación, le recomendamos que vuelva a intentarlo con esta excepción. Para evitar esta excepción, asegúrese de no haber configurado más de 1500 sesiones simultáneas para utilizarlas en un solo libro mayor en todos los clientes. Por ejemplo, puede usar el método maxConcurrentTransactions del controlador Amazon QLDB para Java para configurar el número máximo de sesiones disponibles en una instancia de controlador.

QldbClientException

Mensaje: A streamed result is only valid when the parent transaction is open

La transacción está cerrada y no se puede utilizar para recuperar los resultados de la QLDB. Una transacción se cierra cuando se confirma o se cancela.

Esta excepción se produce cuando el cliente trabaja directamente con el objeto `Transaction` e intenta recuperar los resultados de la QLDB después de haber confirmado o cancelado una transacción. Para mitigar este problema, el cliente debe leer los datos antes de cerrar la transacción.

Exportación de datos de diarios

En esta sección se enumeran las excepciones más comunes que QLDB puede devolver al exportar datos de un diario de un libro mayor a un bucket de Amazon S3. Para obtener más información acerca de esta característica, consulte Exportación de datos de diarios desde Amazon QLDB.

Cada excepción incluye el mensaje de error específico, seguido de una breve descripción y sugerencias de posibles soluciones.

AccessDeniedException

Mensaje: User: *userARN* is not authorized to perform: iam:PassRole on resource: *roleARN*

No tiene permisos para transferir un rol de IAM al servicio de QLDB. QLDB requiere un rol para todas las solicitudes de exportación de diarios y debe tener permisos para transferir este rol a QLDB. El rol proporciona a QLDB permisos de escritura en el bucket de Amazon S3 especificado.

Compruebe que ha definido una política de IAM que conceda permiso para realizar la operación de API `PassRole` en el recurso de rol de IAM especificado para el servicio QLDB (`qldb.amazonaws.com`). Para ver una política de ejemplo, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

IllegalArgumentException

Mensaje: QLDB encountered an error validating S3 configuration: *errorCode errorMessage*

Una posible causa de este error es que el bucket de Amazon S3 proporcionado no existe en Amazon S3. O bien, QLDB no tiene permisos suficientes para escribir objetos en el bucket de Amazon S3 especificado.

Compruebe que el nombre del bucket de S3 que proporcionó en su solicitud de trabajo de exportación es correcto. Para obtener información sobre la nomenclatura de buckets, consulte Restricciones y limitaciones de los buckets en la Guía del usuario de Amazon Simple Storage Service.

Además, compruebe que ha definido una política para el bucket especificado que conceda los permisos `PutObject` y `PutObjectAcl` al servicio QLDB (`qldb.amazonaws.com`). Para obtener más información, consulte Permisos de exportación.

IllegalArgumentException

Mensaje: Unexpected response from Amazon S3 while validating the S3 configuration. Respuesta de S3: *errorCode errorMessage*

El intento de escribir los datos de exportación del diario en el bucket de S3 proporcionado falló con la respuesta de error de Amazon S3 proporcionada. Para obtener más información acerca de las posibles causas, consulte Solución de problemas de Amazon S3 en la Guía de usuario de Amazon Simple Storage Service.

IllegalArgumentException

Mensaje: Amazon S3 bucket prefix must not exceed 128 characters

El prefijo proporcionado en la solicitud de exportación del diario contiene más de 128 caracteres.

IllegalArgumentException

Mensaje: Start date must not be greater than end date

`InclusiveStartTime` y `ExclusiveEndTime` deben estar en formato de fecha y hora ISO 8601 y en hora universal coordinada (UTC).

IllegalArgumentException

Mensaje: End date cannot be in the future

Tanto `InclusiveStartTime` como `ExclusiveEndTime` deben estar en formato de fecha y hora `ISO 8601` y en UTC.

IllegalArgumentException

Mensaje: La configuración de cifrado de objetos proporcionada (s3EncryptionConfiguration) no es compatible con una clave AWS Key Management Service (AWS KMS)

Ha proporcionado un `KMSKeyArn` con un `ObjectEncryptionType` de `NO_ENCRYPTION` o `SSE_S3`. Solo puede proporcionar una AWS KMS key administrada por el cliente para un tipo de cifrado de objeto `SSE_KMS`. Para obtener más información sobre opciones de cifrado del servidor en Amazon S3, consulte protección de datos mediante cifrado del servidor en la Guía del desarrollador de Amazon S3.

LimitExceededException

Mensaje: Exceeded the limit of 2 concurrently running Journal export jobs

QLDB impone un límite predeterminado de dos trabajos de exportación de diarios simultáneos.

Transmisión de datos del diario

En esta sección se enumeran las excepciones más comunes que QLDB puede devolver al transmitir datos de un diario de un libro mayor a Amazon Kinesis Data Streams. Para obtener más información acerca de esta característica, consulte Transmisión de datos de diarios desde Amazon QLDB.

Cada excepción incluye el mensaje de error específico, seguido de una breve descripción y sugerencias de posibles soluciones.

AccessDeniedException

Mensaje: User: *userARN* is not authorized to perform: iam:PassRole on resource: *roleARN*

No tiene permisos para transferir un rol de IAM al servicio de QLDB. QLDB requiere un rol para todas las solicitudes de secuencia del diarios y debe tener permisos para transferir este rol a QLDB. El rol proporciona a QLDB permisos de escritura en el recurso Amazon Kinesis Data Streams especificado.

Compruebe que ha definido una política de IAM que conceda permiso para realizar la operación de API `PassRole` en el recurso de rol de IAM especificado para el servicio QLDB (`qldb.amazonaws.com`). Para ver una política de ejemplo, consulte Ejemplos de políticas basadas en identidades para Amazon QLDB.

IllegalArgumentException

Mensaje: QLDB encountered an error validating Kinesis Data Streams: Respuesta de Kinesis: *errorCode errorMessage*

Una posible causa de este error es que el recurso de Kinesis Data Streams proporcionado no existe. O bien, QLDB no tiene permisos suficientes para escribir registros de datos en el flujo de datos de Kinesis especificado.

Compruebe que el flujo de datos de Kinesis que proporciona en su solicitud de transmisión es correcto. Para obtener más información, consulte Creación y actualización de flujos de datos en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Además, compruebe que ha definido una política para el flujo de datos de Kinesis especificado que conceda al servicio QLDB (`qldb.amazonaws.com`) permisos para las siguientes acciones. Para obtener más información, consulte Permisos de secuencia.

- `kinesis:PutRecord`
- `kinesis:PutRecords`
- `kinesis:DescribeStream`
- `kinesis:ListShards`

IllegalArgumentException

Mensaje: Unexpected response from Kinesis Data Streams while validating the Kinesis configuration. Respuesta de Kinesis: *errorCode errorMessage*

El intento de escribir los registros de datos en el flujo de datos de Kinesis proporcionado falló con la respuesta de error de Kinesis proporcionada. Para obtener más información sobre las posibles causas, consulte Solución de problemas de los productores de Amazon Kinesis Data Streams en la Guía para desarrolladores de Amazon Kinesis Data Streams.

IllegalArgumentException

Mensaje: Start date must not be greater than end date.

`InclusiveStartTime` y `ExclusiveEndTime` deben estar en formato de fecha y hora ISO 8601 y en hora universal coordinada (UTC).

IllegalArgumentException

Mensaje: Start date cannot be in the future.

Tanto `InclusiveStartTime` como `ExclusiveEndTime` deben estar en formato de fecha y hora `ISO 8601` y en UTC.

LimitExceededException

Mensaje: Exceeded the limit of 5 concurrently running Journal streams to Kinesis Data Streams

QLDB impone un límite predeterminado de cinco secuencias de diarios simultáneas.

Verificación de los datos del diario

En esta sección se enumeran las excepciones más comunes que QLDB puede devolver al verificar los datos de un diario de un libro mayor. Para obtener más información acerca de esta característica, consulte Verificación de datos en Amazon QLDB.

Cada excepción incluye el mensaje de error específico, seguido de las operaciones de la API que pueden generarlo, una breve descripción y sugerencias de posibles soluciones.

IllegalArgumentException

Mensaje: The provided Ion value is not valid and cannot be parsed.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

Asegúrese de proporcionar un valor de Amazon Ion válido antes de volver a intentar la solicitud.

IllegalArgumentException

Mensaje: The provided block address is not valid.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

Asegúrese de proporcionar una dirección de bloque válida antes de volver a intentar la solicitud. Una dirección de bloque es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Por ejemplo: `{strandId:"BlFTjlSXze9BIh1K0szcE3",sequenceNo:14}`

IllegalArgumentException

Mensaje: The sequence number of the provided digest tip address is beyond the strand's latest committed record.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

La dirección del tip del resumen que proporcione debe tener un número de secuencia inferior o igual al número de secuencia del último registro confirmado de la cadena del diario. Antes de volver a intentar realizar la solicitud, asegúrese de proporcionar una dirección de tip del resumen con un número de secuencia válido.

IllegalArgumentException

Mensaje: The Strand ID of the provided block address is not valid.

Operaciones de la API: `GetDigest, GetBlock, GetRevision`

La dirección de bloque que proporcione debe tener un identificador de cadena que coincida con el identificador de cadena del diario. Antes de volver a intentar realizar la solicitud, asegúrate de proporcionar una dirección de bloque con un ID de cadena válido.

IllegalArgumentException

Mensaje: The sequence number of the provided block address is beyond the strand's latest committed record.

Operaciones de la API: `GetBlock, GetRevision`

La dirección de bloque que proporcione debe tener un número de secuencia inferior o igual al número de secuencia del último registro confirmado de la cadena. Antes de volver a intentar

realizar la solicitud, asegúrese de proporcionar una dirección de bloque con un número de secuencia válido.

IllegalArgumentException

Mensaje: The Strand ID of the provided block address must match the Strand ID of the provided digest tip address.

Operaciones de la API: `GetBlock, GetRevision`

Solo puede verificar la revisión o el bloque de un documento si existe en la misma cadena del diario que el resumen que ha proporcionado.

IllegalArgumentException

Mensaje: The sequence number of the provided block address must not be greater than the sequence number of the provided digest tip address.

Operaciones de la API: `GetBlock, GetRevision`

Solo puede verificar la revisión o el bloqueo de un documento si está incluido en el resumen que proporcione. Esto significa que se registró en el diario antes que la dirección del tip del resumen.

IllegalArgumentException

Mensaje: The provided Document ID was not found in the block at the specified block address.

Operación de la API: `GetRevision`

El identificador de documento que proporcione debe estar en la dirección de bloque que proporcione. Antes de volver a intentar realizar la solicitud, asegúrese de que estos dos parámetros sean coherentes.

Referencia de PartiQL de Amazon QLDB

Amazon QLDB admite subconjunto del lenguaje de consulta PartiQLhttps://partiql.org/. En las siguientes secciones se describe la implementación QLDB de PartiQL.

> **Note**
>
> - QLDB no es compatible con todas las operaciones de PartiQL.
> - Todas las instrucciones de PartiQL de QLDB están sujetas a límites de transacción, tal como se definen en Cuotas y límites de Amazon QLDB.
> - Esta referencia proporciona sintaxis básica y ejemplos de uso de instrucciones PartiQL que se ejecutan manualmente en la consola QLDB o en el intérprete de comandos QLDB. Para ver ejemplos de código que muestran cómo ejecutar instrucciones similares mediante programación utilizando el controlador de QLDB, consulte los tutoriales en Introducción al controlador.

Temas

¿Qué es PartiQL?

PartiQL proporciona acceso a consultas compatible con SQL en múltiples almacenes de datos que contienen datos estructurados, datos semiestructurados y datos anidados. Es ampliamente utilizado en Amazon y ahora está disponible como parte de muchos Servicios de AWS, incluida QLDB.

Para obtener la especificación PartiQL y un tutorial sobre el lenguaje de consulta principal, consulte la Documentación de PartiQL.

PartiQL amplía el SQL-92 para admitir documentos en el formato de datos de Amazon Ion. Para obtener más información acerca de Amazon Ion, consulte la Referencia del formato de datos de Amazon Ion en Amazon QLDB.

PartiQL en Amazon QLDB

Para ejecutar consultas de PartiQL en QLDB, puede utilizar una de las siguientes opciones:

- El editor de PartiQL en AWS Management Console para QLDB
- El intérprete de comandos de línea de comandos de QLDB
- Un controlador de QLDB proporcionado por AWS para ejecutar consultas mediante programación

Para obtener información sobre el uso de estos métodos para acceder a QLDB, consulte Acceso a Amazon QLDB.

Para obtener información sobre cómo controlar el acceso para ejecutar cada comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Consejos rápidos sobre PartiQL en QLDB

El siguiente es un breve resumen de los consejos y las mejores prácticas para trabajar con PartiQL en QLDB:

- Conozca los límites de concurrencia y de transacciones: todas las instrucciones, incluidas las consultas `SELECT`, están sujetas a un control de concurrencia optimista (OCC), y a límites de transacciones y conflictos, incluido un tiempo de espera de transacción de 30 segundos.
- Utilice índices: utilice índices de cardinalidad alta y ejecute consultas dirigidas para optimizar sus instrucciones y evitar tener que escanear tablas completas. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

- Utilice predicados de igualdad: las búsquedas indexadas requieren un operador de igualdad (= o `IN`). Los operadores de desigualdad (<, >, `LIKE`, `BETWEEN`) no cumplen los requisitos para las búsquedas indexadas y dan como resultado escaneos de tablas completas.
- Utilice únicamente combinaciones internas: QLDB solo admite combinaciones internas. Como práctica recomendada, combine los campos que estén indexados para cada tabla que vaya a unir. Elija índices de cardinalidad alta tanto para los criterios de unión como para los predicados de igualdad.

Convenciones de referencia de PartiQL de Amazon QLDB

En esta sección se explican las convenciones que se utilizan para escribir la sintaxis para las expresiones, los comandos y las funciones PartiQL que se describen en la sección de referencia de PartiQL para Amazon QLDB. Estas convenciones no deben confundirse con la sintaxis y la semántica del propio lenguaje de consultas de PartiQL.

Carácter	Descripción
CAPS	Las palabras en mayúscula son palabras clave.
[ ]	Los corchetes denotan argumentos o cláusulas opcionales. Varios argumentos entre corchetes indican que puede seleccionar cualquier cantidad de argumentos. Además, los argumentos entre corchetes en líneas separadas indican que el analizador de QLDB espera que los argumentos estén en el orden que aparecen en la sintaxis.
|	Las barras verticales indican que puede seleccionar entre los argumentos.
cursiva roja	Las palabras en cursiva roja indican marcadores de posición. Debe insertar el valor adecuado en lugar de la palabra en cursiva roja.
. . .	Los puntos suspensivos indican que puede repetir el elemento anterior.
'	Los valores entre comillas simples indican que debe escribir las comillas. En PartiQL, las comillas simples indican valores de cadena o nombres de campo en las estructuras de Amazon Ion.

Carácter	Descripción
"	Los valores entre comillas dobles indican que debe ingresar las comillas dobles. En PartiQL, las comillas dobles indican los identificadores entre comillas.
`	Los valores entre acentos graves indican que debe ingresar los acentos graves. En PartiQL, los acentos graves indican valores literales de Ion.

Tipos de datos en Amazon QLDB

Los documentos de Amazon QLDB se almacenan en formato Amazon Ion. Amazon Ion es un formato de serialización de datos (tanto en formato de texto como codificado en binario) que es un superconjunto de JSON. En la tabla siguiente se enumeran los tipos de datos Ion que puede usar en documentos QLDB.

Tipo de datos	Descripción
`null`	Un valor null genérico
`bool`	Valores booleanos
`int`	Enteros firmados de tamaño arbitrario
`decimal`	Números reales codificados con decimales de precisión arbitraria
`float`	Números de coma flotante codificados en binario (IEEE de 64 bits)
`timestamp`	Momentos de fecha, hora y zona horaria de precisión arbitraria
`string`	Literales de texto Unicode
`symbol`	Átomos simbólicos de Unicode (identificadores)

Tipo de datos	Descripción
`blob`	Datos binarios de codificación definida por el usuario
`clob`	Datos de texto de codificación definida por el usuario
`struct`	Colecciones desordenadas de pares nombre-valor
`list`	Colecciones de valores heterogéneas ordenadas

Consulte el documento de especificaciones de Ion en el sitio de Amazon GitHub para obtener una lista completa de los tipos de datos principales de Ion con descripciones completas y detalles de formato de valores.

Documentos de Amazon QLDB

Amazon QLDB almacena los registros de datos como documentos, que son simplemente objetos `struct` de Amazon Ion que se insertan en una tabla. Para ver la especificación de Ion, consulte el sitio GitHub de Amazon Ion.

Temas

- Estructura del documento de Ion
- Mapeo de tipo PartiQL-Ion
- ID del documento

Estructura del documento de Ion

Al igual que JSON, los documentos de QLDB se componen de pares nombre-valor en la siguiente estructura.

```
{
  name1: value1,
```

```
  name2: value2,
  name3: value3,
  ...
  nameN: valueN
}
```

Los nombres son un token de símbolo y los valores no tienen restricciones. Cada par nombre-valor se denomina campo. El valor de un campo puede ser cualquiera de los Tipos de datos de Ion, incluidos los tipos de contenedores: estructuras anidadas, listas y listas de estructuras.

Al igual que en JSON, se indica `struct` con paréntesis (`{...}`) y `list` se indica con corchetes (`[...]`). El siguiente ejemplo es un documento de los datos de muestra de Introducción a la consola de Amazon QLDB que contiene valores de diversos tipos.

```
{
    VIN: "1N4AL11D75C109151",
    LicensePlateNumber: "LEWISR261LL",
    State: "WA",
    City: "Seattle",
    PendingPenaltyTicketAmount: 90.25,
    ValidFrom: 2017-08-21T,
    ValidTo: 2020-05-11T,
    Owners: {
        PrimaryOwner: { PersonId: "294jJ3YUoH1IEEm8GSabOs" },
        SecondaryOwners: [{ PersonId: "5Ufgdlnj06gF5CWcOIu64s" }]
    }
}
```

Important

En Ion, las comillas dobles indican valores de cadena y los símbolos sin comillas representan nombres de campo. Pero en PartiQL, las comillas simples indican tanto las cadenas como los nombres de los campos.

Esta sintaxis permite que el lenguaje de consultas de PartiQL mantenga la compatibilidad con SQL y que el formato de datos Amazon Ion mantenga la compatibilidad con JSON. Para obtener más información sobre la sintaxis y la semántica de PartiQL en QLDB, consulte Consulta de Ion con PartiQL.

Mapeo de tipo PartiQL-Ion

En QLDB, PartiQL amplía el sistema de tipos de SQL para cubrir el modelo de datos de Ion. Este mapeo se describe de la siguiente manera:

- Los escalares de tipo SQL están cubiertos por sus homólogos de Ion. Por ejemplo:
  - `CHAR` y `VARCHAR` son secuencias Unicode que se asocian al tipo `string` de Ion.
  - `NUMBER` se asocia al tipo `decimal` de Ion.
- El tipo `struct` de Ion equivale a una tupla de SQL, que tradicionalmente representa una fila de una tabla.
  - Sin embargo, con contenido abierto y sin esquema, no se admiten consultas que se basen en la naturaleza ordenada de una tupla de SQL (por ejemplo, el orden de salida de `SELECT *`).
- Además de `NULL`, PartiQL tiene un tipo `MISSING`. Esta es una especialización de `NULL` e indica la falta de un campo. Este tipo es necesario porque los campos `struct` de Ion pueden ser dispersos.

ID del documento

QLDB asigna un identificador de documento a cada documento que se inserta en una tabla. Todos los identificadores asignados por el sistema son identificadores únicos universales (UUID), cada uno de los cuales se representa en una cadena codificada en Base62 (por ejemplo, `3Qv67yjXEwB9SjmvkuG6Cp`). Para obtener más información, consulte Identificadores únicos en Amazon QLDB.

Cada revisión de documento se identifica de forma única mediante una combinación del identificador del documento y un número de versión de base cero.

Los campos ID de documento y versión se incluyen en los metadatos del documento, que puede consultar en la vista confirmada (la vista de una tabla definida por el sistema). Para obtener más información acerca de las vistas en QLDB, consulte Conceptos clave. Para obtener más información sobre metadatos, consulte Consulta de los metadatos del documento.

Consulta de Ion con PartiQL en Amazon QLDB

Cuando consulta datos en Amazon QLDB, escribe instrucciones en formato PartiQL, pero QLDB devuelve los resultados en formato Amazon Ion. PartiQL está pensado para ser compatible con SQL,

mientras que Ion es una extensión de JSON. Esto genera diferencias sintácticas entre la forma de anotar los datos en las instrucciones de consulta y la forma en que se muestran los resultados de la consulta.

En esta sección se describen la sintaxis y la semántica básicas para ejecutar instrucciones de PartiQL manualmente mediante la consola de QLDB o el intérprete de comandos de QLDB.

> **Tip**
>
> Al ejecutar consultas de PartiQL mediante programación, la mejor práctica es utilizar instrucciones parametrizadas. Puede utilizar un signo de interrogación (?) como marcador de posición de una variable de enlace en sus instrucciones para evitar estas reglas de sintaxis. Esto también es más seguro y eficiente.
> Para obtener más información, consulte los siguientes tutoriales en introducción al controlador:
>
> - Java: Tutorial de inicio rápido | Referencia de libro de recetas
> - .NET: Tutorial de inicio rápido | Referencia de libro de recetas
> - Go: Tutorial de inicio rápido | Referencia de libro de recetas
> - Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
> - Python: Tutorial de inicio rápido | Referencia de libro de recetas

Temas

Sintaxis y semántica

Cuando se utiliza la consola de QLDB o el intérprete de comandos de QLDB para consultar datos de Ion, la sintaxis y la semántica fundamentales de PartiQL son las siguientes:

Sensibilidad de mayúsculas y minúsculas

Todos los nombres de objetos del sistema de QLDB, incluidos los nombres de campos, tablas y libros mayores, distinguen mayúsculas de minúsculas.

Valores de cadena

En Ion, las comillas dobles (`"..."`) indican una cadena.

En PartiQL, las comillas simples (`'...'`) indican una cadena.

Símbolos e identificadores

En Ion, las comillas simples (`'...'`) indican un símbolo. En Ion, un subconjunto de símbolos denominados identificadores se representa mediante texto sin comillas.

En PartiQL, las comillas dobles (`"..."`) indican identificadores PartiQL entre comillas, como una palabra reservada que se usa como nombre de tabla. El texto sin comillas representa un identificador de PartiQL normal, como el nombre de una tabla que no es una palabra reservada.

Literales de Ion

Cualquier literal de Ion se puede indicar con acentos graves (`` `...` ``) en una instrucción de PartiQL.

Nombres de los campos

Todos los nombres de campo en Ion distinguen entre mayúsculas y minúsculas. PartiQL permite indicar los nombres de los campos con comillas simples en una instrucción DML. Esta es una alternativa abreviada al uso de la función `cast` de PartiQL para definir un símbolo. También es más intuitivo que usar acentos graves para denotar un símbolo literal de Ion.

Literales

Los literales del lenguaje de consulta PartiQL corresponden a los tipos de datos de Ion, de la siguiente manera:

Escalares

Siga la sintaxis SQL cuando proceda, como se describe en la sección Mapeo de tipo PartiQL-Ion. Por ejemplo:

- 5

- `'foo'`
- `null`

Structs

También se conocen como tuplas u objetos en muchos formatos y otros modelos de datos.

Se indican mediante paréntesis (`{...}`) con los elementos `struct` separados por comas.

- `{ 'id' : 3, 'arr': [1, 2] }`

Lists

También se conocen como matrices.

Se indican mediante paréntesis (`[...]`) con los elementos de lista separados por comas.

- `[ 1, 'foo' ]`

Bags

Colecciones desordenadas en PartiQL.

Se indican mediante corchetes de doble ángulo (`<<...>>`) con los elementos de lista separados por comas. En QLDB, una tabla se puede concebir como un bag. Sin embargo, no se puede anidar un bag dentro de los documentos de una tabla.

- `<< 1, 'foo' >>`

Ejemplo

A continuación, se muestra un ejemplo de la sintaxis de una instrucción `INSERT` con diversos tipos de Ion.

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
```

```
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidFromDate' : `2017-09-14T`, --Ion timestamp literal with day precision
    'ValidToDate' : `2020-06-25T`
}
```

Notación con acentos graves

PartiQL cubre todos los tipos de datos de Ion, por lo que puede escribir cualquier instrucción sin usar acentos graves. Sin embargo, hay casos en los que esta sintaxis literal de Ion puede hacer que sus instrucciones sean más claras y concisas.

Por ejemplo, para insertar un documento con valores de marca de tiempo y símbolo de Ion, puede escribir la siguiente instrucción utilizando únicamente la sintaxis de PartiQL.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': to_timestamp('2019-09-04T'),
    'mySymbol': cast('foo' as symbol)
}
```

Esto es bastante detallado, por lo que, en su lugar, puede utilizar acentos graves para simplificar la instrucción.

```
INSERT INTO myTable VALUE
{
    'myTimestamp': `2019-09-04T`,
    'mySymbol': `foo`
}
```

También puede incluir toda la estructura entre acentos graves para ahorrar algunas pulsaciones del teclado más.

```
INSERT INTO myTable VALUE
`{
    myTimestamp: 2019-09-04T,
    mySymbol: foo
}`
```

Important

Las cadenas y los símbolos son clases diferentes en PartiQL. Esto significa que, aunque tengan el mismo texto, no son iguales. Por ejemplo, las siguientes expresiones de PartiQL se evalúan en valores de Ion diferentes.

```
'foo'
```

```
`foo`
```

Navegación de rutas

Al escribir un lenguaje de manipulación de datos (DML) o instrucciones de consulta, puede acceder a los campos dentro de estructuras anidadas mediante pasos de ruta. PartiQL es compatible con la notación de puntos, que permite acceder a los nombres de campo de una estructura principal. En el siguiente ejemplo se accede al campo `Model` de un `Vehicle` principal.

```
Vehicle.Model
```

Para acceder a un elemento específico de una lista, puede utilizar el operador de corchetes para indicar un número ordinal de base cero. En el ejemplo siguiente se accede al elemento de `SecondaryOwners` con un número ordinal de 2. En otras palabras, este es el tercer elemento de la lista.

```
SecondaryOwners[2]
```

Uso de alias

QLDB admite contenido y esquema abiertos. Por lo tanto, cuando accede a campos concretos de una instrucción, la mejor manera de asegurarse de obtener los resultados esperados es utilizar alias. Por ejemplo, si no especifica un alias explícito, el sistema generará uno implícito para sus orígenes `FROM`.

```
SELECT VIN FROM Vehicle
--is rewritten to
SELECT Vehicle.VIN FROM Vehicle AS Vehicle
```

Sin embargo, los resultados son impredecibles en caso de conflictos de nombres de campo. Si existe otro campo con el nombre `VIN` en una estructura anidada dentro de los documentos, los valores `VIN` devueltos por esta consulta pueden sorprenderle. Como práctica recomendada, escriba la siguiente instrucción en su lugar. Esta consulta declara `v` como un alias que se extiende por encima de la tabla `Vehicle`. La palabra clave `AS` es opcional.

```
SELECT v.VIN FROM Vehicle [ AS ] v
```

El uso de alias resulta especialmente útil cuando se crea una ruta a colecciones anidadas dentro de un documento. Por ejemplo, la siguiente instrucción declara `o` como un alias que abarca toda la colección `VehicleRegistration.Owners`.

```
SELECT o.SecondaryOwners
FROM VehicleRegistration AS r, @r.Owners AS o
```

En este caso, el carácter `@` es técnicamente opcional. Sin embargo, indica de forma explícita que desea la estructura `Owners` dentro de `VehicleRegistration` y no una recopilación diferente denominada `Owners` (si existiera).

Especificación de PartiQL

Para obtener más información acerca del lenguaje de consultas de PartiQL, consulte la especificación de PartiQL.

Comandos de PartiQL en Amazon QLDB

PartiQL amplía el SQL-92 para admitir documentos en el formato de datos de Amazon Ion. Amazon QLDB admite los siguientes comandos de PartiQL.

Para obtener información sobre cómo controlar el acceso para ejecutar cada comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

> **Note**
>
> - QLDB no es compatible con todos los comandos de PartiQL.
> - Todas las instrucciones de PartiQL de QLDB están sujetas a límites de transacción, tal como se definen en Cuotas y límites de Amazon QLDB.

- Esta referencia proporciona sintaxis básica y ejemplos de uso de instrucciones PartiQL que se ejecutan manualmente en la consola QLDB o en el intérprete de comandos QLDB. Para ver ejemplos de código que muestran cómo ejecutar instrucciones utilizando un lenguaje de programación compatible, consulte los tutoriales en Introducción al controlador.

Instrucciones en lenguaje de definición de datos (DDL)

El lenguaje de definición de datos (DDL) es el conjunto de instrucciones de PartiQL que se utiliza para administrar objetos de bases de datos, como tablas e índices. El DDL se utiliza para crear y eliminar estos objetos.

- CREATE INDEX
- CREATE TABLE
- DROP INDEX
- DROP TABLE
- UNDROP TABLE

Instrucciones de lenguaje de manipulación de datos (DML)

Lenguaje de manipulación de datos(DML) es el conjunto de instrucciones PartiQL que se utiliza para administrar datos en tablas de QLDB. Puede usar instrucciones DML para agregar, modificar o eliminar datos de una tabla.

Se admiten las siguientes instrucciones DML y lenguaje de consulta:

- DELETE
- FROM (INSERT, REMOVE o SET)
- INSERT
- SELECT
- UPDATE

Comando CREATE INDEX en Amazon QLDB

En Amazon QLDB, utilice el comando `CREATE INDEX` para crear un índice para un campo de documento de una tabla.

Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

> **Important**
>
> QLDB requiere un índice para buscar un documento de manera eficiente. Sin un índice, QLDB necesita escanear toda la tabla al leer los documentos. Esto puede provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
>
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad (`=` o `IN`) en un campo indexado o en un ID de documento. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Tenga en cuenta las siguientes restricciones al crear índices:

- Solo se puede crear un índice en un único campo de nivel superior. No se admiten índices compuestos, anidados, únicos ni basados en funciones.
- Puede crear un índice en cualquier tipo de datos de Ion, incluidos `list` y `struct`. Sin embargo, solo puede realizar la búsqueda indexada igualando el valor total de Ion, independientemente del tipo de Ion. Por ejemplo, cuando se utiliza un tipo `list` como índice, no se puede realizar una búsqueda indexada por un elemento de la lista.
- El rendimiento de las consultas solo mejora cuando se utiliza un predicado de igualdad; por ejemplo, `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`.

  QLDB no respeta las desigualdades en los predicados de consulta. Como resultado, no se implementan los escaneos filtrados por rango.
- Los nombres de los campos indexados distinguen entre mayúsculas y minúsculas y pueden tener 128 caracteres como máximo.
- La creación de índices en QLDB es asíncrona. La cantidad de tiempo que tarda en crearse un índice en una tabla que no está vacía varía según el tamaño de la tabla. Para obtener más información, consulte Administrar índices.

Temas

- Sintaxis

Sintaxis

```
CREATE INDEX ON table_name (field)
```

Parámetros

table_name

Nombre de la tabla en la que desea crear el índice. La tabla debe existir previamente.

El nombre de la tabla distingue entre mayúsculas y minúsculas.

campo

El nombre del campo del documento para el que se va a crear el índice. El campo debe ser un atributo de nivel superior.

Los nombres de los campos indexados distinguen entre mayúsculas y minúsculas y pueden tener 128 caracteres como máximo.

Puede crear un índice en cualquier tipo de datos de Amazon Ion, incluidos `list` y `struct`. Sin embargo, solo puede realizar la búsqueda indexada igualando el valor total de Ion, independientemente del tipo de Ion. Por ejemplo, cuando se utiliza un tipo `list` como índice, no se puede realizar una búsqueda indexada por un elemento de la lista.

Valor devuelto

`tableId`: el identificador único de la tabla en la que creó el índice.

Ejemplos

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

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

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas
- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando CREATE TABLE en Amazon QLDB

En Amazon QLDB, utilice el comando `CREATE TABLE` para crear una tabla nueva.

Las tablas tienen nombres simples que no tienen espacios de nombres. QLDB admite contenido abierto y no aplica el esquema, por lo que no se definen atributos o tipos de datos al crear tablas.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en el libro mayor, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

```
CREATE TABLE table_name [ WITH (aws_tags = `{'key': 'value'}`) ]
```

Parámetros

table_name

Nombre de la tabla única que se va a crear. No debe existir ya una tabla activa con el mismo nombre. Estas son las restricciones de denominación:

- Las etiquetas solo deben contener de 1 a 128 caracteres alfanuméricos o guiones bajos.
- Debe tener una letra o un guion bajo para el primer carácter.
- Puede contener cualquier combinación de caracteres alfanuméricos y guiones bajos para el resto de los caracteres.
- Distingue entre mayúsculas y minúsculas.
- No debe ser una palabra reservada para QLDB PartiQL.

'***key***': '***value***'

(Opcional) Las etiquetas que se deben adjuntar al recurso de tabla durante la creación. Cada etiqueta se define como un par clave-valor, donde la clave y el valor se indican cada uno entre comillas simples. Cada par clave-valor se define dentro de una estructura de Amazon Ion que se indica con acentos graves.

Actualmente, el etiquetado de tablas al crearlas solo se admite en los libros mayores en el modo de permisos *STANDARD*.

Valor devuelto

`tableId`: el identificador único de la tabla que creó.

Etiquetado de tablas al crearlas

> **Note**
>
> Actualmente, el etiquetado de tablas al crearlas solo se admite en los libros mayores en el modo de permisos `STANDARD`.

También puede etiquetar los recursos de la tabla especificando las etiquetas en una instrucción `CREATE TABLE`. Para obtener más información acerca de las etiquetas, consulte Etiquetado de recursos de Amazon QLDB. En el siguiente ejemplo, se crea una tabla llamada `Vehicle` con la etiqueta `environment=production`.

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'production'}`)
```

Para etiquetar las tablas al crearlas, es necesario acceder a las acciones `qldb:PartiQLCreateTable` y `qldb:TagResource`. Para obtener más información acerca de los permisos para los recursos de QLDB, consulte Cómo funciona Amazon QLDB con IAM.

Al etiquetar los recursos en el momento de su creación, ya no es necesario ejecutar scripts de etiquetado personalizados después de la creación del recurso. Una vez etiquetada una tabla, puede controlar el acceso a la tabla según esas etiquetas. Por ejemplo, puede conceder acceso total solo a las tablas que tengan una etiqueta específica. Para ver una política de ejemplo JSON, consulte Acceso completo a todas las acciones basadas en las etiquetas de las tablas.

Ejemplos

```
CREATE TABLE VehicleRegistration
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'environment': 'development'}`)
```

```
CREATE TABLE Vehicle WITH (aws_tags = `{'key1': 'value1', 'key2': 'value2'}`)
```

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas
- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando DELETE en Amazon QLDB

En Amazon QLDB, utilice el comando `DELETE` para marcar un documento activo como eliminado en una tabla mediante la creación de una revisión nueva, pero definitiva del documento. Esta última revisión indica que el documento se ha eliminado. Esta operación finaliza el ciclo de vida de un documento, lo que significa que no se pueden crear más revisiones del documento con el mismo identificador de documento.

La operación es irreversible. Aún puede consultar el historial de revisiones de un documento eliminado utilizando Función de historial.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

```
DELETE FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
```

Parámetros

table_name

Nombre de la tabla de usuario que contiene los datos que se van a eliminar. Las instrucciones de DML solo se admiten en la vista de usuario predeterminada. Cada instrucción solo puede ejecutarse en una sola tabla.

AS *table_alias*

(Opcional) Un alias definido por el usuario que se extiende a lo largo de una tabla de la que se va a eliminar. La palabra clave `AS` es opcional.

BY *id_alias*

(Opcional) Un alias definido por el usuario que se enlaza con el campo `id` de metadatos de cada documento del conjunto de resultados. El alias debe declararse en la cláusula `FROM` mediante la palabra clave `BY`. Esto resulta útil cuando se desea filtrar por identificador del documento al consultar la vista de usuario predeterminada. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

WHERE *condition*

Criterios de selección para los documentos que se van a eliminar.

> **Note**
>
> Si omite la cláusula `WHERE`, se eliminarán todos los elementos de la tabla.

Valor devuelto

`documentId`: el identificador único de cada documento que ha eliminado.

Ejemplos

```
DELETE FROM VehicleRegistration AS r
WHERE r.VIN = '1HVBBAANXWH544237'
```

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas
- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas

- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando DROP INDEX en Amazon QLDB

En Amazon QLDB, utilice el comando `DROP INDEX` para eliminar un índice de una tabla.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

```
DROP INDEX "indexId" ON table_name WITH (purge = true)
```

> **Note**
>
> La cláusula `WITH (purge = true)` es obligatoria para todas las instrucciones `DROP INDEX` y actualmente `true` es el único valor admitido.
> La palabra clave `purge` distingue entre mayúsculas y minúsculas y debe escribirse completamente en minúsculas.

Parámetros

"*indexId*"

El identificador único del índice que se va a eliminar, indicado entre comillas dobles.

ON *table_name*

Nombre de la tabla cuyo índice desea eliminar.

Valor devuelto

`tableId`: el identificador único de la tabla cuyo índice ha eliminado.

Ejemplos

```
DROP INDEX "4tPW3fUhaVhDinRgKRLhGU" ON VehicleRegistration WITH (purge = true)
```

Comando DROP TABLE en Amazon QLDB

En Amazon QLDB, utilice el comando `DROP TABLE` para desactivar una tabla existente. Puede utilizar la instrucción UNDROP TABLE para reactivarla. La desactivación o reactivación de una tabla no afecta a sus documentos o índices.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

```
DROP TABLE table_name
```

Parámetros

table_name

Nombre de la tabla que se va a desactivar. Esta tabla debe existir y tener un estado de `ACTIVE`.

Valor devuelto

`tableId`: el identificador exclusivo de la tabla que ha desactivado.

Ejemplos

```
DROP TABLE VehicleRegistration
```

Comando FROM (INSERT, REMOVE o SET) en Amazon QLDB

En Amazon QLDB, una instrucción que comienza con `FROM` es una extensión PartiQL que permite insertar y eliminar elementos específicos de un documento. También puede usar esta instrucción para actualizar los elementos existentes en un documento, de forma similar al comando UPDATE.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

FROM-INSERT

Inserte un elemento nuevo en un documento existente. Para insertar un documento nuevo de nivel superior en una tabla, debe utilizar INSERT.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
INSERT INTO element VALUE data [ AT key_name ]
```

FROM-REMOVE

Elimine un elemento existente de un documento o elimine todo un documento de nivel superior. Esta última es semánticamente la misma que la sintaxis DELETE tradicional.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
REMOVE element
```

FROM-SET

Actualice uno o más elementos de un documento. Si un elemento no existe, se inserta. Esta es semánticamente la misma que la sintaxis UPDATE tradicional.

```
FROM table_name [ AS table_alias ] [ BY id_alias ]
[ WHERE condition ]
SET element = data [, element = data, ... ]
```

Parámetros

table_name

Nombre de la tabla de usuario que contiene los datos que se van a modificar. Las instrucciones de DML solo se admiten en la vista de usuario predeterminada. Cada instrucción solo puede ejecutarse en una sola tabla.

En esta cláusula, también puede incluir una o más colecciones anidadas en la tabla especificada. Para obtener más información, consulte Colecciones anidadas.

AS *table_alias*

(Opcional) Un alias definido por el usuario que se extiende a lo largo de una tabla que se va a modificar. Todos los alias de tabla que se utilizan en la cláusula `SET`, `REMOVE`, `INSERT INTO` o `WHERE` deben declararse en la cláusula `FROM`. La palabra clave `AS` es opcional.

BY *id_alias*

(Opcional) Un alias definido por el usuario que se enlaza con el campo `id` de metadatos de cada documento del conjunto de resultados. El alias debe declararse en la cláusula `FROM` mediante la palabra clave `BY`. Esto resulta útil cuando se desea filtrar por identificador del documento al consultar la vista de usuario predeterminada. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

WHERE *condition*

Criterios de selección para los documentos que se van a modificar.

> **Note**
>
> Si omite la cláusula `WHERE`, se modificarán todos los documentos de la tabla.

element

Elemento del documento que se va a crear o modificar.

data

Un nuevo valor para el elemento.

AT *key_name*

Un nombre clave que se añadirá a los documentos que se van a modificar. Debe especificar el `VALUE` correspondiente junto con el nombre de la clave. Esto es necesario para insertar un nuevo valor `AT` en una posición específica dentro de un documento.

Colecciones anidadas

Si bien puede ejecutar una instrucción DML solo en una tabla, puede especificar colecciones anidadas dentro de los documentos de esa tabla como orígenes adicionales. Cada alias que declare para una colección anidada se puede usar en la cláusula `WHERE` y en la cláusula `SET`, `INSERT INTO` o `REMOVE`.

Por ejemplo, los orígenes `FROM` de la siguiente instrucción incluyen tanto la tabla `VehicleRegistration` como la estructura `Owners.SecondaryOwners` anidada.

```
FROM VehicleRegistration r, @r.Owners.SecondaryOwners o
WHERE r.VIN = '1N4AL11D75C109151' AND o.PersonId = 'abc123'
SET o.PersonId = 'def456'
```

En este ejemplo se actualiza el elemento específico de la lista `SecondaryOwners` que tiene un `PersonId` de `'abc123'` dentro del documento `VehicleRegistration` que tiene un `VIN` de `'1N4AL11D75C109151'`. Esta expresión permite especificar un elemento de una lista por su valor en lugar de por su índice.

Valor devuelto

`documentId`: el identificador único de cada documento que ha actualizado o eliminado.

Ejemplos

Modificar un elemento de un documento. Si un elemento no existe, se inserta.

```
FROM Vehicle AS v
WHERE v.VIN = '1N4AL11D75C109151' AND v.Color = 'Silver'
SET v.Color = 'Shiny Gray'
```

Modificar o insertar un elemento y filtrar en el campo de metadatos del `id` del documento asignado por el sistema.

```
FROM Vehicle AS v BY v_id
WHERE v_id = 'documentId'
SET v.Color = 'Shiny Gray'
```

Modificar el campo `PersonId` del primer elemento de la lista `Owners.SecondaryOwners` dentro de un documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

Insertar un elemento existente en un documento.

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

Eliminar un documento completo de una tabla.

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

Eliminar el primer elemento de la lista `Owners.SecondaryOwners` dentro de un documento en la tabla `VehicleRegistration`.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
REMOVE r.Owners.SecondaryOwners[0]
```

Insertar `{'Mileage':26500}` como un par nombre-valor de nivel superior dentro de un documento de la tabla `Vehicle`.

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

Añadir `{'PersonId':'abc123'}` como un par nombre-valor en el campo `Owners.SecondaryOwners` de un documento de la tabla `VehicleRegistration`. Tenga en cuenta que `Owners.SecondaryOwners` debe existir ya y debe ser un tipo de datos de lista para que esta instrucción sea válida. De lo contrario, la palabra clave `AT` es obligatoria en la cláusula `INSERT INTO`.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
```

Insertar `{'PersonId':'abc123'}` como primer elemento de la lista `Owners.SecondaryOwners` existente dentro de un documento.

```
FROM VehicleRegistration AS r
```

```
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
```

Añadir varios pares de nombre-valor a la lista `Owners.SecondaryOwners` existente dentro de un documento.

```
FROM VehicleRegistration AS r
WHERE r.VIN = '1N4AL11D75C109151'
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
```

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas
- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando INSERT en Amazon QLDB

En Amazon QLDB, utilice el comando `INSERT` para añadir uno o más documentos de Amazon Ion a una tabla.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

- Sintaxis
- Parámetros

- Valor devuelto
- Ejemplos
- Ejecución mediante programación con el controlador

Sintaxis

Insertar un documento.

```
INSERT INTO table_name VALUE document
```

Insertar varios documentos.

```
INSERT INTO table_name << document, document, ... >>
```

Parámetros

table_name

Nombre de la tabla en la que desea insertar los datos. La tabla debe existir previamente. Las instrucciones de DML solo se admiten en la vista de usuario predeterminada.

documento

Un documento QLDB válido. Debe especificar al menos un documento. Los múltiples documentos deben ir separados por comas.

El documento debe indicarse con llaves (`{...}`).

Cada nombre de campo del documento es un símbolo Ion que distingue entre mayúsculas y minúsculas y que se puede indicar con comillas simples (`'...'`) en PartiQL.

Los valores de cadena también se denotan con comillas simples (`'...'`) en PartiQL.

Cualquier literal de Ion se puede indicar con acentos graves (`` `...` ``).

> **Note**
>
> Los corchetes de doble ángulo (`<<...>>`) indican una colección desordenada (conocida como bag en PartiQL) y solo son necesarios si desea insertar múltiples documentos.

Valor devuelto

`documentId`: el identificador único de cada documento que ha insertado.

Ejemplos

Insertar un documento.

```
INSERT INTO VehicleRegistration VALUE
{
    'VIN' : 'KM8SRDHF6EU074761', --string
    'RegNum' : 1722, --integer
    'State' : 'WA',
    'City' : 'Kent',
    'PendingPenaltyTicketAmount' : 130.75, --decimal
    'Owners' : { --nested struct
        'PrimaryOwner' : { 'PersonId': '294jJ3YUoH1IEEm8GSabOs' },
        'SecondaryOwners' : [ --list of structs
            { 'PersonId' : '1nmeDdLo3AhGswBtyM1eYh' },
            { 'PersonId': 'IN7MvYtUjkp1GMZu0F6CG9' }
        ]
    },
    'ValidFromDate' : `2017-09-14T`, --Ion timestamp literal with day precision
    'ValidToDate' : `2020-06-25T`
}
```

Esta instrucción devuelve el identificador único del documento que ha insertado, de la siguiente manera.

```
{
    documentId: "2kKuOPNB07D2iTPBrUTWGl"
}
```

Insertar varios documentos.

```
INSERT INTO Person <<
{
    'FirstName' : 'Raul',
    'LastName' : 'Lewis',
    'DOB' : `1963-08-19T`,
    'GovId' : 'LEWISR261LL',
    'GovIdType' : 'Driver License',
```

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

Esta instrucción devuelve el identificador único de cada documento que ha insertado, de la siguiente manera.

```
{
    documentId: "6WXzLscsJ3bDWW97Dy8nyp"
},
{
    documentId: "35e0ToZyTGJ7LGvcwrkX65"
},
{
    documentId: "BVHPcH612o7JROQ4yP8jiH"
}
```

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas

- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando SELECT en Amazon QLDB

En Amazon QLDB, utilice el comando `SELECT` para recuperar datos de una o más tablas. En QLDB, cada consulta `SELECT` se procesa en una transacción y está sujeta a un límite de tiempo de espera de la transacción.

El orden de los resultados no es específico y puede variar para cada consulta `SELECT`. No debe confiar en el orden de los resultados para ninguna consulta en QLDB.

Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

> **Warning**
>
> Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. PartiQL admite este tipo de consultas porque es compatible con SQL. Sin embargo, no ejecute escaneados de tablas para casos de uso de producción en QLDB. Los escaneados de tablas pueden provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad en un campo indexado o en un ID de documento, por ejemplo `WHERE indexedField = 123` o `WHERE indexedField IN (456, 789)`. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Temas

- Sintaxis
- Parámetros
- combinaciones;
- Limitaciones de consultas anidadas
- Ejemplos

- Ejecución mediante programación con el controlador

Sintaxis

```
SELECT [ VALUE ] expression [ AS field_alias ] [, expression, ... ]
FROM source [ AS source_alias ] [ AT idx_alias ] [ BY id_alias ] [, source, ... ]
[ WHERE condition ]
```

Parámetros

VALUE

Un calificador para la expresión que hace que la consulta devuelva el valor del tipo de datos sin procesar, en lugar de incluir el valor en una estructura de tupla.

expression

Una proyección formada a partir del comodín * o una lista de proyección de uno o más campos de documentos del conjunto de resultados. Una expresión puede consistir en llamadas a Funciones de PartiQL o campos modificados por Operadores PartiQL.

AS ***field_alias***

(Opcional) Un alias temporal definido por el usuario para el campo que se utiliza en el conjunto de resultados finales. La palabra clave AS es opcional.

Si no se especifica un alias para una expresión que no sea un nombre de campo simple, el conjunto de resultados aplica un nombre predeterminado a ese campo.

FROM ***source***

Un origen que se va a consultar. Los únicos orígenes admitidos actualmente son los nombres de las tablas, las combinaciones internas entre tablas, las consultas SELECT anidadas (sujetas a Limitaciones de consultas anidadas) y las llamadas a las funciones del historial de una tabla.

Debe especificar al menos un origen. Los múltiples orígenes deben ir separados por comas.

AS ***source_alias***

(Opcional) Un alias definido por el usuario que se extiende a lo largo de del origen que se va a consultar. Todos los alias de origen que se utilizan en las cláusulas SELECT O WHERE deben declararse en la cláusula FROM. La palabra clave AS es opcional.

AT *idx_alias*

(Opcional) Un alias definido por el usuario que enlaza con el número de índice (ordinal) de cada elemento de una lista del origen. El alias debe declararse en la cláusula `FROM` mediante la palabra clave `AT`.

BY *id_alias*

(Opcional) Un alias definido por el usuario que se enlaza con el campo `id` de metadatos de cada documento del conjunto de resultados. El alias debe declararse en la cláusula `FROM` mediante la palabra clave `BY`. Esto resulta útil cuando se desea proteger o filtrar por identificador del documento al consultar la vista de usuario predeterminada. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

WHERE *condition*

Los criterios de selección y los criterios de combinación (si procede) de la consulta.

> **Note**
>
> Si omite la cláusula `WHERE`, se recuperarán todos los elementos de la tabla.

combinaciones;

Actualmente solo se admiten combinaciones internas. Puede escribir consultas de combinación internas mediante la cláusula `INNER JOIN` explícita, de la siguiente manera. En esta sintaxis, `JOIN` debe combinarse con `ON` y la palabra clave `INNER` es opcional.

```
SELECT expression
FROM table1 AS t1 [ INNER ] JOIN table2 AS t2
ON t1.element = t2.element
```

O bien, puede escribir combinaciones internas mediante la sintaxis implícita, de la siguiente manera.

```
SELECT expression
FROM table1 AS t1, table2 AS t2
WHERE t1.element = t2.element
```

Limitaciones de consultas anidadas

Puede escribir consultas anidadas (subconsultas) dentro de las expresiones `SELECT` y los orígenes `FROM`. La principal restricción es que solo la consulta más externa puede acceder al entorno de base de datos global. Por ejemplo, suponga que tiene un libro mayor con tablas `VehicleRegistration` y `Person`. La siguiente consulta anidada no es válida porque `SELECT` interno intenta acceder a `Person`.

```
SELECT r.VIN,
    (SELECT p.PersonId FROM Person AS p WHERE p.PersonId =
 r.Owners.PrimaryOwner.PersonId) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Mientras que la siguiente consulta anidada es válida.

```
SELECT r.VIN, (SELECT o.PrimaryOwner.PersonId FROM @r.Owners AS o) AS PrimaryOwner
FROM VehicleRegistration AS r
```

Ejemplos

La siguiente consulta muestra un comodín básico `SELECT` todos con una cláusula de predicado `WHERE` estándar que utiliza el operador `IN`.

```
SELECT * FROM Vehicle
WHERE VIN IN ('1N4AL11D75C109151', 'KM8SRDHF6EU074761')
```

A continuación se muestran las proyecciones `SELECT` con un filtro de cadena.

```
SELECT FirstName, LastName, Address
FROM Person
WHERE Address LIKE '%Seattle%'
AND GovId = 'LEWISR261LL'
```

En el siguiente ejemplo, se muestra una subconsulta correlacionada que aplana datos anidados. En este caso, el carácter `@` es técnicamente opcional. Sin embargo, indica de forma explícita que desea la estructura `Owners` que está anidada en `VehicleRegistration`, y no una recopilación diferente denominada `Owners` (si existiera). Para obtener más contexto, consulte Datos anidados en el capítulo Cómo trabajar con datos e historial.

```
SELECT
```

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

A continuación, se muestra una subconsulta de la lista SELECT que proyecta datos anidados, además de una combinación interna implícita.

```
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')
```

A continuación, se muestra una combinación interna explícita.

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

A continuación, se muestra una proyección del campo de metadatos `id` del documento, utilizando la cláusula BY.

```
SELECT
    r_id,
    r.VIN
FROM
    VehicleRegistration AS r BY r_id
WHERE
```

```
    r_id = 'documentId'
```

A continuación, se utiliza la cláusula BY para combinar las tablas `DriversLicense` y `Person` en sus campos `PersonId` y `id` del documento, respectivamente.

```
SELECT * FROM DriversLicense AS d INNER JOIN Person AS p BY pid
ON d.PersonId = pid
WHERE pid = 'documentId'
```

A continuación, se utiliza Vista confirmada para combinar las tablas `DriversLicense` y `Person` en sus campos `PersonId` y `id` del documento, respectivamente.

```
SELECT * FROM DriversLicense AS d INNER JOIN _ql_committed_Person AS cp
ON d.PersonId = cp.metadata.id
WHERE cp.metadata.id = 'documentId'
```

Lo siguiente devuelve el `PersonId` y el número de índice (ordinal) de cada persona de la lista `Owners.SecondaryOwners` de un documento de la tabla `VehicleRegistration`.

```
SELECT s.PersonId, owner_idx
FROM VehicleRegistration AS r, @r.Owners.SecondaryOwners AS s AT owner_idx
WHERE r.VIN = 'KM8SRDHF6EU074761'
```

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas
- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando UPDATE en Amazon QLDB

En Amazon QLDB, utilice el comando `UPDATE` para modificar el valor de uno o más elementos dentro de un documento. Si un elemento no existe, se inserta.

También puede usar este comando para insertar y eliminar de forma explícita elementos específicos de un documento, de forma similar a las instrucciones FROM (INSERT, REMOVE o SET).

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

UPDATE-SET

Actualice uno o más elementos de un documento. Si un elemento no existe, se inserta. Semánticamente es lo mismo que la instrucción FROM-SET.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
SET element = data [, element = data, ... ]
[ WHERE condition ]
```

UPDATE-INSERT

Inserte un elemento nuevo en un documento existente. Para insertar un documento nuevo de nivel superior en una tabla, debe utilizar INSERT.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
INSERT INTO element VALUE data [ AT key_name ]
[ WHERE condition ]
```

UPDATE-REMOVE

Elimine un elemento existente de un documento o elimine todo un documento de nivel superior. Esta última es semánticamente la misma que la sintaxis DELETE tradicional.

```
UPDATE table_name [ AS table_alias ] [ BY id_alias ]
REMOVE element
[ WHERE condition ]
```

Parámetros

table_name

Nombre de la tabla de usuario que contiene los datos que se van a modificar. Las instrucciones de DML solo se admiten en la vista de usuario predeterminada. Cada instrucción solo puede ejecutarse en una sola tabla.

AS ***table_alias***

(Opcional) Un alias definido por el usuario que se extiende a lo largo de una tabla de la que se va a actualizar. La palabra clave `AS` es opcional.

BY ***id_alias***

(Opcional) Un alias definido por el usuario que se enlaza con el campo de metadatos `id` de cada documento del conjunto de resultados. El alias debe declararse en la cláusula `UPDATE` mediante la palabra clave `BY`. Esto resulta útil cuando se desea filtrar por identificador del documento al consultar la vista de usuario predeterminada. Para obtener más información, consulte Uso de la cláusula BY para consultar el identificador del documento.

element

Elemento del documento que se va a crear o modificar.

data

Un nuevo valor para el elemento.

AT ***key_name***

Un nombre clave que se añadirá a los documentos que se van a modificar. Debe especificar el `VALUE` correspondiente junto con el nombre de la clave. Esto es necesario para insertar un nuevo valor `AT` en una posición específica dentro de un documento.

WHERE ***condition***

Criterios de selección para los documentos que se van a modificar.

> **Note**
>
> Si omite la cláusula `WHERE`, se modificarán todos los documentos de la tabla.

Valor devuelto

`documentId`: el identificador único de cada documento que ha actualizado.

Ejemplos

Actualizar un campo de un documento. Si un campo no existe, se inserta.

```
UPDATE Person AS p
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE p.GovId = '111-22-3333'
```

Filtrar por el campo de metadatos del `id` de documento asignado por el sistema.

```
UPDATE Person AS p BY pid
SET p.LicenseNumber = 'HOLLOR123ZZ'
WHERE pid = 'documentId'
```

Sobrescribir un documento completo.

```
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'
```

Modificar el campo `PersonId` del primer elemento de la lista `Owners.SecondaryOwners` dentro de un documento.

```
UPDATE VehicleRegistration AS r
SET r.Owners.SecondaryOwners[0].PersonId = 'abc123'
```

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

Insertar `{'Mileage':26500}` como un par nombre-valor de nivel superior dentro de un documento de la tabla `Vehicle`.

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

Añadir `{'PersonId':'abc123'}` como un par nombre-valor en el campo `Owners.SecondaryOwners` de un documento de la tabla `VehicleRegistration`. Tenga en cuenta que `Owners.SecondaryOwners` debe existir ya y debe ser un tipo de datos de lista para que esta instrucción sea válida. De lo contrario, la palabra clave `AT` es obligatoria en la cláusula `INSERT INTO`.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE { 'PersonId' : 'abc123' }
WHERE r.VIN = '1N4AL11D75C109151'
```

Insertar `{'PersonId':'abc123'}` como primer elemento de la lista `Owners.SecondaryOwners` existente dentro de un documento.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners VALUE {'PersonId' : 'abc123'} AT 0
WHERE r.VIN = '1N4AL11D75C109151'
```

Añadir varios pares de nombre-valor a la lista `Owners.SecondaryOwners` existente dentro de un documento.

```
UPDATE VehicleRegistration AS r
INSERT INTO r.Owners.SecondaryOwners << {'PersonId' : 'abc123'}, {'PersonId' :
 'def456'} >>
WHERE r.VIN = '1N4AL11D75C109151'
```

Insertar un elemento existente en un documento.

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

Eliminar un documento completo de una tabla.

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

Eliminar el primer elemento de la lista `Owners.SecondaryOwners` dentro de un documento en la tabla `VehicleRegistration`.

```
UPDATE VehicleRegistration AS r
REMOVE r.Owners.SecondaryOwners[0]
WHERE r.VIN = '1N4AL11D75C109151'
```

Ejecución mediante programación con el controlador

Para aprender a ejecutar esta instrucción mediante programación con el controlador de QLDB, consulte los siguientes tutoriales en Introducción al controlador:

- Java: Tutorial de inicio rápido | Referencia de libro de recetas
- .NET: Tutorial de inicio rápido | Referencia de libro de recetas
- Go: Tutorial de inicio rápido | Referencia de libro de recetas
- Node.js: Tutorial de inicio rápido | Referencia de libro de recetas
- Python: Tutorial de inicio rápido | Referencia de libro de recetas

Comando UNDROP TABLE en Amazon QLDB

En Amazon QLDB, utilice el comando `UNDROP TABLE` para reactivar una tabla que eliminó anteriormente (es decir, desactivó). La desactivación o reactivación de una tabla no afecta a sus documentos o índices.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

Sintaxis

```
UNDROP TABLE "tableId"
```

Parámetros

"*tableId*"

El identificador único de la tabla que se va a reactivar, indicado entre comillas dobles.

La tabla debe haberse eliminado previamente, lo que significa que existe en la tabla del catálogo del sistema `information_schema.user_tables` y su estado es `INACTIVE`. Tampoco debe haber ninguna tabla activa existente con el mismo nombre.

Valor devuelto

`tableId`: el identificador único de la tabla reactivó.

Ejemplos

```
UNDROP TABLE "5PLf9SXwndd63lPaSIa0O6"
```

Funciones de PartiQL en Amazon QLDB

PartiQL en Amazon QLDB admite las siguientes variantes integradas de funciones estándar de SQL.

> **Note**
>
> QLDB no admite actualmente ninguna función SQL que no se incluya en esta lista.
> Esta referencia de funciones se basa en la documentación de PartiQL Funciones integradas.

Propagación de tipo desconocido (nulo y ausente)

A menos que se indique lo contrario, todas estas funciones propagan valores de argumentos nulos y ausentes. La propagación de `NULL` o `MISSING` se define como el retorno de `NULL` si algún argumento de la función es `NULL` o `MISSING`. A continuación se muestran algunos ejemplos de esta propagación.

```
CHAR_LENGTH(null)    -- null
CHAR_LENGTH(missing) -- null (also returns null)
```

Aggregate functions (Funciones de agregación)

- AVG
- COUNT
- MAX
- MIN
- SIZE
- SUM

Funciones condicionales

- COALESCE
- EXISTS
- NULLIF

Funciones de fecha y hora

- DATE_ADD
- DATE_DIFF
- EXTRACT
- UTCNOW

Funciones escalares

- TXID

Funciones de cadena

- CHAR_LENGTH
- CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

Funciones de formato de tipo de datos

- CAST
- TO_STRING
- TO_TIMESTAMP

Función AVG en Amazon QLDB

En Amazon QLDB, utilice la función `AVG` para devolver el promedio (media aritmética) de los valores de la expresión de entrada. Esta función funciona con valores numéricos e ignora los valores nulos o faltantes.

Sintaxis

```
AVG ( expression )
```

Argumentos

`expression`

El nombre de campo o la expresión de un tipo de datos numérico sobre el que opera la función.

Tipos de datos

Tipos de argumentos admitidos:

- `int`
- `decimal`
- `float`

Tipo de devolución: `decimal`

Ejemplos

```
SELECT AVG(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 147.19
SELECT AVG(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 2.
```

Funciones relacionadas

- COUNT
- MAX
- MIN
- SIZE
- SUM

Función CAST en Amazon QLDB

En Amazon QLDB, use la función `CAST` para evaluar una expresión determinada en un valor y convertir el valor en un tipo de datos de destino específico. Si la conversión no puede realizarse, la función devolverá un error.

Sintaxis

```
CAST ( expression AS type )
```

Argumentos

`expression`

El nombre del campo o expresión que se evalúa para obtener el valor que la función convierte. La conversión de valores nulos devuelve valores nulos. Este parámetro puede ser cualquiera de los Tipos de datos compatibles.

type

El nombre del tipo de datos de destino para la conversión. Este parámetro puede ser uno de los Tipos de datos compatibles.

Tipo de retorno

El tipo de datos especificado por el argumento *type*.

Ejemplos

Los siguientes ejemplos muestran la propagación de tipos desconocidos (NULL o MISSING).

```
CAST(null    AS null)    -- null
CAST(missing AS null)    -- null
CAST(missing AS missing) -- missing
CAST(null    AS missing) -- missing
CAST(null    AS boolean) -- null (null AS any data type name results in null)
CAST(missing AS boolean) -- missing (missing AS any data type name results in missing)
```

Los valores que no sean de un tipo desconocido no se pueden convertir en NULL oMISSING.

```
CAST(true AS null)    -- error
CAST(true AS missing) -- error
CAST(1    AS null)    -- error
CAST(1    AS missing) -- error
```

En los siguientes ejemplos se muestra la conversión 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'`)
```

En los siguientes ejemplos se muestra la conversión 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
```

En los siguientes ejemplos se muestra la conversión `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
```

En los siguientes ejemplos se muestra la conversión `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
```

En los siguientes ejemplos se muestra la conversión `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
```

En los siguientes ejemplos se muestra la conversión `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'
```

En los siguientes ejemplos se muestra la conversión `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"
```

En los siguientes ejemplos se muestra la conversión `AS struct`.

```
CAST(`{ a: 1 }` AS struct) -- {a:1}
CAST(true       AS struct) -- err
```

En los siguientes ejemplos se muestra la conversión `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
```

Los siguientes ejemplos son instrucciones ejecutables que incluyen algunos de los ejemplos anteriores.

```
SELECT CAST(true AS integer) FROM << 0 >>            -- 1
SELECT CAST('2001-01-01T' AS timestamp) FROM << 0 >> -- 2001-01-01T
SELECT CAST('xx' AS symbol) FROM << 0 >>             -- xx
SELECT CAST(42 AS string) FROM << 0 >>               -- "42"
```

Funciones relacionadas

- TO_STRING
- TO_TIMESTAMP

Función CHAR_LENGTH en Amazon QLDB

En Amazon QLDB, use la función `CHAR_LENGTH` para devolver el número de caracteres de la cadena especificada, donde character se define como un único punto de código Unicode.

Sintaxis

```
CHAR_LENGTH ( string )
```

`CHAR_LENGTH` es sinónimo de Función CHARACTER_LENGTH en Amazon QLDB.

Argumentos

`string`

El nombre de campo o la expresión del tipo de datos `string` que evalúa la función.

Tipo de retorno

`int`

Ejemplos

```
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)
```

Funciones relacionadas

- LOWER
- SUBSTRING
- TRIM
- UPPER

Función CHARACTER_LENGTH en Amazon QLDB

Sinónimo de la función `CHAR_LENGTH`.

Consulte Función CHAR_LENGTH en Amazon QLDB.

Función COALESCE en Amazon QLDB

En Amazon QLDB, dada una lista de uno o más argumentos, use la función `COALESCE` para evaluar los argumentos en orden de izquierda a derecha y devolver el primer valor que no sea de tipo desconocido (`NULL` o `MISSING`). Si todos los tipos de argumentos son desconocidos, el resultado es `NULL`.

La función `COALESCE` no propaga `NULL` ni `MISSING`.

Sintaxis

```
COALESCE ( expression [, expression, ... ] )
```

Argumentos

`expression`

La lista de uno o varios nombres de campo o expresiones que evalúa la función. Cada argumento puede ser cualquiera de los Tipos de datos compatibles.

Tipo de retorno

Cualquier tipo de datos compatible. El tipo de retorno es o bien `NULL` o bien el tipo de la primera expresión que se evalúa como un valor no nulo y no faltante.

Ejemplos

```
SELECT COALESCE(1, null) FROM << 0 >>          -- 1
SELECT COALESCE(null, null, 1) FROM << 0 >>    -- 1
SELECT COALESCE(null, 'string') FROM << 0 >>   -- "string"
```

Funciones relacionadas

- EXISTS
- NULLIF

Función COUNT en Amazon QLDB

En Amazon QLDB, use la función `COUNT` para devolver el número de documentos definidos por la expresión dada. Esta función tiene dos variaciones:

- `COUNT(*)`: cuenta todos los documentos de la tabla de destino, incluyan o no valores nulos o faltantes.
- `COUNT(expression)`: calcula el número de documentos con valores no nulos en un campo o expresión existente específico.

> **Warning**
>
> La función `COUNT` no está optimizada, por lo que no recomendamos usarla sin una búsqueda indexada. Cuando ejecuta una consulta en QLDB sin una búsqueda indexada, se invoca un escaneo completo de la tabla. Esto puede provocar problemas de rendimiento en tablas grandes, como conflictos de concurrencia y tiempos de espera de las transacciones.
> Para evitar el escaneado de tablas, debe ejecutar las instrucciones con una cláusula de predicado `WHERE` usando un operador de igualdad (`=` o `IN`) en un campo indexado o en un ID de documento. Para obtener más información, consulte Optimizar el rendimiento de las consultas.

Sintaxis

```
COUNT ( * | expression )
```

Argumentos

expression

El nombre de campo o la expresión sobre la que opera la función. Este parámetro puede ser cualquiera de los Tipos de datos compatibles.

Tipo de retorno

`int`

Ejemplos

```
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
```

Funciones relacionadas

- AVG
- MAX
- MIN
- SIZE
- SUM

Función DATE_ADD en Amazon QLDB

En Amazon QLDB, use la función `DATE_ADD` para incrementar un valor de marca de tiempo determinado en un intervalo específico.

Sintaxis

```
DATE_ADD( datetimepart, interval, timestamp )
```

Argumentos

datetimepart

La parte de la fecha o la hora sobre la que opera la función. Este parámetro puede ser uno de los siguientes:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

interval

El número entero que especifica el intervalo a añadir a la *marca de tiempo* dada. Un número entero negativo resta al intervalo.

timestamp

El nombre de campo o la expresión del tipo de datos `timestamp` que incrementa la función.

El valor literal de una marca de tiempo de Ion se puede indicar con comillas invertidas (`` `...` ``). Para obtener más detalles sobre formatos y ejemplos de valores de marcas de tiempo, consulte Marcas de tiempo en el documento de especificaciones de Amazon Ion.

Tipo de retorno

`timestamp`

Ejemplos

```
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
```

Funciones relacionadas

- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Función DATE_DIFF en Amazon QLDB

En Amazon QLDB, use la función `DATE_DIFF` para devolver la diferencia entre las partes de fecha especificadas de dos marcas de tiempo determinadas.

Sintaxis

```
DATE_DIFF( datetimepart, timestamp1, timestamp2 )
```

Argumentos

`datetimepart`

La parte de la fecha o la hora sobre la que opera la función. Este parámetro puede ser uno de los siguientes:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`

timestamp1, *timestamp2*

Los dos nombres de cambo o expresiones del tipo de datos `timestamp` que compara la función. Si *timestamp2* es posterior a *timestamp1*, el resultado es positivo. Si *timestamp2* es anterior a *timestamp1*, el resultado es negativo.

El valor literal de una marca de tiempo de Ion se puede indicar con comillas invertidas (`` `...` ``). Para obtener más detalles sobre formatos y ejemplos de valores de marcas de tiempo, consulte Marcas de tiempo en el documento de especificaciones de Amazon Ion.

Tipo de retorno

`int`

Ejemplos

```
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
```

Funciones relacionadas

- DATE_ADD
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Función EXISTS en Amazon QLDB

En Amazon QLDB, dado un valor de PartiQL, use la función `EXISTS` para devolver `TRUE` si el valor es una colección no vacía. De lo contrario, esta función devolverá `FALSE`. Si la entrada de `EXISTS` no es un contenedor, el resultado es `FALSE`.

La función `EXISTS` no propaga `NULL` ni `MISSING`.

Sintaxis

```
EXISTS ( value )
```

Argumentos

`value`

El nombre de campo o la expresión que evalúa la función. Este parámetro puede ser cualquiera de los Tipos de datos compatibles.

Tipo de retorno

`bool`

Ejemplos

```
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
```

Funciones relacionadas

- COALESCE
- NULLIF

Función EXTRACT en Amazon QLDB

En Amazon QLDB, use la función `EXTRACT` para devolver el valor entero de una fecha especificada o la parte de hora de una marca de tiempo determinada.

Sintaxis

```
EXTRACT ( datetimepart FROM timestamp )
```

Argumentos

`datetimepart`

La parte de fecha u hora que extrae la función. Este parámetro puede ser uno de los siguientes:

- `year`
- `month`
- `day`
- `hour`
- `minute`
- `second`
- `timezone_hour`
- `timezone_minute`

`timestamp`

El nombre de campo o la expresión del tipo de datos `timestamp` del que se extrae la función. Si este parámetro es de tipo desconocido (`NULL` o `MISSING`), la función devuelve `NULL`.

El valor literal de una marca de tiempo de Ion se puede indicar con comillas invertidas (`` `...` ``). Para obtener más detalles sobre formatos y ejemplos de valores de marcas de tiempo, consulte Marcas de tiempo en el documento de especificaciones de Amazon Ion.

Tipo de retorno

`int`

Ejemplos

```
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
```

Funciones relacionadas

- DATE_ADD
- DATE_DIFF
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

Función LOWER en Amazon QLDB

En Amazon QLDB, use la función `LOWER` para convertir todos los caracteres de una determinada cadena de mayúsculas a minúsculas.

Sintaxis

```
LOWER ( string )
```

Argumentos

string

El nombre de campo o la expresión del tipo de datos `string` que convierte la función.

Tipo de retorno

`string`

Ejemplos

```
SELECT LOWER('AbCdEfG!@#$') FROM << 0 >> -- 'abcdefg!@#$'
```

Funciones relacionadas

- CHAR_LENGTH
- SUBSTRING
- TRIM
- UPPER

Función MAX en Amazon QLDB

En Amazon QLDB, use la función `MAX` para devolver el valor máximo de un conjunto de valores numéricos.

Sintaxis

```
MAX ( expression )
```

Argumentos

expression

El nombre de campo o la expresión de un tipo de datos numérico sobre el que opera la función.

Tipos de datos

Tipos de argumentos admitidos:

- `int`
- `decimal`
- `float`

Tipos de devolución admitidos:

- `int`
- `decimal`
- `float`

Ejemplos

```
SELECT MAX(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 442.30
SELECT MAX(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 3
```

Funciones relacionadas

- AVG
- COUNT
- MIN
- SIZE
- SUM

Función MIN en Amazon QLDB

En Amazon QLDB, use la función `MIN` para devolver el valor mínimo de un conjunto de valores numéricos.

Sintaxis

```
MIN ( expression )
```

Argumentos

expression

El nombre de campo o la expresión de un tipo de datos numérico sobre el que opera la función.

Tipos de datos

Tipos de argumentos admitidos:

- `int`
- `decimal`
- `float`

Tipos de devolución admitidos:

- `int`
- `decimal`
- `float`

Ejemplos

```
SELECT MIN(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 30.45
SELECT MIN(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 1
```

Funciones relacionadas

- AVG
- COUNT
- MAX
- SIZE
- SUM

Función NULLIF en Amazon QLDB

En Amazon QLDB, dadas dos expresiones, use la función `NULLIF` para devolver `NULL` si ambas evalúan el mismo valor. En caso contrario, la función devuelve el resultado de la evaluación de la primera expresión.

La función `NULLIF` no propaga `NULL` ni `MISSING`.

Sintaxis

```
NULLIF ( expression1, expression2 )
```

Argumentos

`expression1`, *`expression2`*

Los dos nombres de campo o expresiones que compara la función. Estos parámetros pueden ser cualquiera de los Tipos de datos admitidos.

Tipo de retorno

Cualquier tipo de datos compatible. El tipo de devolución es `NULL` o el mismo que el tipo de la primera expresión.

Ejemplos

```
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
```

Funciones relacionadas

- COALESCE
- EXISTS

Función SIZE en Amazon QLDB

En Amazon QLDB, use la función `SIZE` para devolver el número de elementos de un determinado tipo de datos de contenedor (lista, estructura o bolsa).

Sintaxis

```
SIZE ( container )
```

Argumentos

contenedor

El nombre de campo del contenedor o la expresión sobre la que opera la función.

Tipos de datos

Tipos de argumentos admitidos:

- Lista
- estructura
- bolsa

Tipo de devolución: `int`

Si la entrada de `SIZE` no es un contenedor, la función arroja un error.

Ejemplos

```
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
```

Funciones relacionadas

- AVG
- COUNT
- MAX
- MIN
- SUM

Función SUBSTRING en Amazon QLDB

En Amazon QLDB, use la función `SUBSTRING` para devolver una subcadena de una cadena determinada. La subcadena comienza en el índice de inicio especificado y finaliza en el último carácter de la cadena o en la longitud especificada.

Sintaxis

```
SUBSTRING ( string, start-index [, length ] )
```

Argumentos

`string`

El nombre del campo o la expresión del tipo de datos `string` del que se va a extraer una subcadena.

`índice de comienzo`

La posición inicial dentro de la *`cadena`* desde la que comenzar la extracción. Este número puede ser negativo.

El primer carácter de la *`cadena`* tiene un índice de 1.

longitud

(Opcional) El número de caracteres (puntos de código) a extraer de la *cadena*, comenzando en *start-index* y finalizando en (*start-index* + *length*) - 1. Es decir, la longitud de la subcadena. Este número no puede ser negativo.

Si no se proporciona este parámetro, la función continúa hasta el final de la *cadena*.

Tipo de retorno

`string`

Ejemplos

```
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"
```

Funciones relacionadas

- CHAR_LENGTH
- LOWER
- TRIM
- UPPER

Función SUM en Amazon QLDB

En Amazon QLDB, use la función SUM para devolver la suma de los valores del campo de entrada o de la expresión. Esta función funciona con valores numéricos e ignora los valores nulos o faltantes.

Sintaxis

```
SUM ( expression )
```

Argumentos

expression

El nombre de campo o la expresión de un tipo de datos numérico sobre el que opera la función.

Tipos de datos

Tipos de argumentos admitidos:

- `int`
- `decimal`
- `float`

Tipos de devolución admitidos:

- `int`: para argumentos de números enteros
- `decimal`: para argumentos decimales o de punto flotante

Ejemplos

```
SELECT SUM(r.PendingPenaltyTicketAmount) FROM VehicleRegistration r  -- 735.95
SELECT SUM(a) FROM << { 'a' : 1 }, { 'a': 2 }, { 'a': 3 } >>         -- 6
```

Funciones relacionadas

- AVG

- COUNT
- MAX
- MIN
- SIZE

Función TO_STRING en Amazon QLDB

En Amazon QLDB, use la función `TO_STRING` para devolver una representación en cadena de una marca de tiempo determinada en el patrón de formato especificado.

Sintaxis

```
TO_STRING ( timestamp, 'format' )
```

Argumentos

`timestamp`

El nombre de campo o la expresión de un tipo de datos `timestamp` que convierte la función en una cadena.

El valor literal de una marca de tiempo de Ion se puede indicar con comillas invertidas (`` `...` ``). Para obtener más detalles sobre formatos y ejemplos de valores de marcas de tiempo, consulte Marcas de tiempo en el documento de especificaciones de Amazon Ion.

`formato`

La cadena literal que especifica el patrón de formato del resultado, en términos de sus partes de fecha. Para conocer los formatos válidos, consulte Cadenas con formato de marca de tiempo.

Tipo de retorno

`string`

Ejemplos

```
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"
```

Funciones relacionadas

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_TIMESTAMP
- UTCNOW

Función TO_TIMESTAMP en Amazon QLDB

En Amazon QLDB, dada una cadena que representa una marca de tiempo, use la función `TO_TIMESTAMP` para convertir la cadena en un tipo de datos `timestamp`. Esta es la operación inversa de `TO_STRING`.

Sintaxis

```
TO_TIMESTAMP ( string [, 'format' ] )
```

Argumentos

`string`

El nombre de campo o la expresión de un tipo de datos `string` que la función convierte en una marca temporal.

`formato`

(Opcional) El literal de cadena que define el patrón del formato de la entrada *`cadena`*, en términos de sus partes de fecha. Para conocer los formatos válidos, consulte Cadenas con formato de marca de tiempo.

Si se omite este argumento, la función asume que la *`cadena`* tiene el formato de una marca de tiempo estándar de Ion. Esta es la manera recomendada de analizar una marca de tiempo de Ion mediante esta función.

El relleno con cero es opcional cuando se emplea un símbolo de formato de un solo carácter (como `y`, `M`, `d`, `H`, `h`, `m`, `s`), pero es obligatorio para sus variantes con relleno de ceros (como `yyyy`, `MM`, `dd`, `HH`, `hh`, `mm`, `ss`).

Se da un tratamiento especial a los años de dos dígitos (símbolo de formato `yy`). 1900 se suma a valores superiores o iguales a 70, y 2000, a valores inferiores a 70.

Los nombres de los meses y los indicadores AM y PM no distinguen entre mayúsculas y minúsculas.

Tipo de retorno

`timestamp`

Ejemplos

```
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
```

Funciones relacionadas

- CAST
- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- UTCNOW

Función TRIM en Amazon QLDB

En Amazon QLDB, use la función `TRIM` para recortar una determinada cadena eliminando los espacios en blanco iniciales y finales o un conjunto específico de caracteres.

Sintaxis

```
TRIM ( [ LEADING | TRAILING | BOTH [ characters ] FROM ] string)
```

Argumentos

LEADING

(Opcional) Indica que los espacios en blanco o los caracteres especificados deben eliminarse del principio de la *cadena*. Si no se especifica, el comportamiento predeterminado es `BOTH`.

TRAILING

(Opcional) Indica que los espacios en blanco o los caracteres especificados deben eliminarse del final de la *cadena*. Si no se especifica, el comportamiento predeterminado es `BOTH`.

BOTH

(Opcional) Indica que tanto los espacios en blanco al principio y al final, como los caracteres especificados deben eliminarse del principio y al final de la *cadena*.

caracteres

(Opcional) El conjunto de caracteres que se va a eliminar, especificado como `string`.

Si no se proporciona este parámetro, se eliminan los espacios en blanco.

string

El nombre del campo o la expresión del tipo de datos `string` que recorta la función.

Tipo de retorno

`string`

Ejemplos

```
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         "
```

Funciones relacionadas

- CHAR_LENGTH
- LOWER
- SUBSTRING
- UPPER

Función TXID en Amazon QLDB

En Amazon QLDB, use la función `TXID` para devolver la ID de transacción única de la instrucción actual que está ejecutando. Es el valor que se asigna al campo de metadatos `txId` de un documento cuando la transacción actual se registra en el diario.

Sintaxis

```
TXID()
```

Argumentos

Ninguno

Tipo de retorno

`string`

Ejemplos

```
SELECT TXID() FROM << 0 >>  -- "L7S9iJqcn9W2M4qOEn27ay"
SELECT TXID() FROM Person WHERE GovId = 'LEWISR261LL'  -- "BKeMb48PNyvHWJGZHkaodG"
```

Función UPPER en Amazon QLDB

En Amazon QLDB, use la función UPPER para convertir todos los caracteres de una determinada cadena de minúsculas a mayúsculas.

Sintaxis

```
UPPER ( string )
```

Argumentos

`string`

El nombre de campo o la expresión del tipo de datos `string` que convierte la función.

Tipo de retorno

`string`

Ejemplos

```
SELECT UPPER('AbCdEfG!@#$') FROM << 0 >> -- 'ABCDEFG!@#$'
```

Funciones relacionadas

- CHAR_LENGTH
- LOWER
- SUBSTRING
- TRIM

Función UTCNOW en Amazon QLDB

En Amazon QLDB, use la función `UTCNOW` para devolver la hora actual en formato de Hora Universal Coordinada (UTC) como `timestamp`.

Sintaxis

```
UTCNOW()
```

Esta función requiere que especifique una cláusula `FROM` en una consulta `SELECT`.

Argumentos

Ninguno

Tipo de retorno

`timestamp`

Ejemplos

```
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'
```

Funciones relacionadas

- DATE_ADD
- DATE_DIFF

- EXTRACT
- TO_STRING
- TO_TIMESTAMP

Cadenas con formato de marca de tiempo

En esta sección se proporciona información de referencia para las cadenas con formato de marca de tiempo.

Las cadenas con formato de marca de tiempo se aplican a las funciones `TO_STRING` y `TO_TIMESTAMP`. Estas cadenas pueden contener separadores de partes de fecha (como "`-`", "`/`" o "`:`") y los siguientes símbolos de formato.

Formato	Ejemplo	Descripción
yy	70	Año de dos dígitos
y	1970	Los cuatro dígitos del año
yyyy	1970	Año en cuatro dígitos rellenado con ceros
M	1	Mes como número entero
MM	01	Valor entero del mes rellenado con ceros
MMM	Jan	Nombre del mes abreviado
MMMM	Enero	Nombre completo del mes
d	2	Día del mes (1-31)
dd	02	Día del mes rellenado con ceros (01-31)
a	AM o PM	Indicador meridiano (para formato de 12 horas)

Formato	Ejemplo	Descripción
h	3	Hora (formato de 12 horas, de 01 a 12)
hh	03	Hora con relleno de ceros (formato de 12 horas, de 01 a 12)
H	3	Hora (formato de 24 horas, de 00 a 23)
HH	03	Hora con relleno cero (formato de 24 horas, de 00 a 23)
m	4	Minutos (00 a 59)
mm	04	Minutos con relleno de ceros (de 00 a 59)
s	5	Segundos (00 a 59)
ss	05	Segundos con relleno de ceros (de 00 a 59)
S	0	Fracción de un segundo (precisión: 0,1, rango: 0,0-0,9)
SS	06	Fracción de un segundo (precisión: 0,01, rango: 0,0-0,99)
SSS	060	Fracción de un segundo (precisión: 0,001, rango: 0,0-0,999)
X	+07 o Z	Desplazamiento respecto s UTC en horas o "Z" si el desplazamiento es 0

Formato	Ejemplo	Descripción
XX	+0700 o Z	Desplazamiento respecto a UTC en horas y minutos o "Z" si el desplazamiento es 0
XXX	+07:00 o Z	Desplazamiento respecto a UTC en horas y minutos o "Z" si el desplazamiento es 0
x	+07	Desfase respecto a UTC en horas
xx	+0700	Desplazamiento respecto a UTC en horas y minutos
xxx	+07:00	Desplazamiento respecto a UTC en horas y minutos

Procedimientos almacenados PartiQL en Amazon QLDB

En Amazon QLDB, puede usar el comando `EXEC` para ejecutar procedimientos almacenados PartiQL con la siguiente sintaxis.

```
EXEC stored_procedure_name argument [, ... ]
```

QLDB es compatible solo con los siguientes procedimientos almacenados del sistema:

Temas

- Procedimiento almacenado REDACT_REVISION en Amazon QLDB

Procedimiento almacenado REDACT_REVISION en Amazon QLDB

> **Note**
>
> Cualquier libro mayor que se haya creado antes del 22 de julio de 2021 actualmente no es apto para la edición. Puede ver la hora de creación de su libro mayor en la consola de Amazon QLDB.

En Amazon QLDB, utilice el procedimiento almacenado `REDACT_REVISION` para eliminar permanentemente una revisión de documento individual e inactiva tanto en el almacenamiento indexado como en el almacenamiento del diario. Este procedimiento almacenado elimina todos los datos de usuario de la revisión especificada. Sin embargo, no modifica la secuencia del diario y los metadatos del documento, incluidos el identificador del documento y el hash. La operación es irreversible.

La revisión del documento especificada debe ser una revisión inactiva en el historial. La última revisión activa de un documento no es apta para ser editada.

Cuando envía una solicitud de edición ejecutando el procedimiento almacenado, la QLDB procesa la edición de los datos de forma asíncrona. Una vez finalizada la edición, los datos de usuario de la revisión especificada (representados por la estructura `data`) se sustituyen por un nuevo campo `dataHash`. El valor de este campo es el hash de Amazon Ion de la estructura `data` eliminada. Como resultado, el libro mayor mantiene la integridad general de sus datos y sigue siendo verificable criptográficamente mediante las operaciones de la API de verificación existentes.

Para ver un ejemplo de una operación de edición con datos de muestra, consulte Ejemplo de edición en Editar revisiones de documentos.

> **Note**
>
> Para obtener información sobre cómo controlar el acceso para ejecutar este comando PartiQL en tablas específicas, consulte Introducción al modo de permisos estándar en Amazon QLDB.

Temas

- Condiciones y limitaciones de edición

- Sintaxis
- Argumentos
- Valor devuelto
- Ejemplos

Condiciones y limitaciones de edición

Antes de empezar con la edición de datos en Amazon QLDB, asegúrese de revisar las siguientes consideraciones y limitaciones:

- El procedimiento almacenado `REDACT_REVISION` se centra en los datos de usuario en una revisión de documento individual e inactiva. Para editar varias revisiones, debe ejecutar el procedimiento almacenado una vez para cada revisión. Puede editar una revisión por transacción.
- Para editar campos concretos de una revisión de un documento, primero debe utilizar una instrucción de lenguaje de manipulación de datos (DML) independiente para modificar la revisión. Para obtener más información, consulte Editar un campo concreto de una revisión.
- Una vez que QLDB reciba una solicitud de edición, no podrá cancelarla ni modificarla. Para confirmar si una edición está completa, puede comprobar si la estructura `data` de la revisión se ha sustituido por un campo `dataHash`. Para obtener más información, consulte Comprobar si una edición está completa.
- La edición no afecta a ningún dato de QLDB que se replique fuera del servicio de QLDB. Esto incluye cualquier exportación a Amazon S3 y las transmisiones a Amazon Kinesis Data Streams. Debe utilizar otros métodos de retención de datos para gestionar los datos almacenados fuera de la QLDB.
- La edición no afecta a los valores literales de las instrucciones PartiQL que se registran en el diario. Como práctica recomendada, se deben ejecutar instrucciones parametrizadas mediante programación utilizando marcadores de posición de variables en lugar de valores literales. Los marcadores de posición se escriben en el diario como un signo de interrogación (?) y no como información confidencial que pueda requerir edición.

  Para aprender a ejecutar instrucciones PartiQL mediante programación con el controlador QLDB, consulte los tutoriales de cada lenguaje de programación compatible en Introducción al controlador.

Sintaxis

```
EXEC REDACT_REVISION `block-address`, 'table-id', 'document-id'
```

Argumentos

`` `block-address` ``

La ubicación en el bloque del diario de la revisión del documento que se va a editar. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Se trata de un valor literal de Ion que se indica con acentos graves. Por ejemplo:

```
`{strandId:"JdxjkR9bSYB5jMHWcI464T", sequenceNo:17}`
```

Para obtener información sobre cómo encontrar la dirección del bloque, consulte Consulta de los metadatos del documento.

'*table-id*'

El identificador único de la tabla cuya revisión del documento desea editar, indicado entre comillas simples.

Para obtener información sobre cómo encontrar el identificador de la tabla, consulte Consulta del catálogo del sistema.

'*document-id*'

El identificador único del documento cuya revisión desea editar, indicado entre comillas simples.

Para obtener información sobre cómo encontrar el identificador del documento, consulte Consulta de los metadatos del documento.

Valor devuelto

Una estructura de Amazon Ion que representa la revisión del documento que se va a editar, en el siguiente formato.

```
{
  blockAddress: {
    strandId: String,
    sequenceNo: Int
```

```
  },
  tableId: String,
  documentId: String,
  version: Int
}
```

Devolver los campos de estructura

- `blockAddress`: la ubicación en el bloque del diario de la revisión que se va a editar. Una dirección tiene los dos campos siguientes:
  - `strandId`: el identificador único de la cadena del diario que contiene el bloque.
  - `sequenceNo`: un número de índice que especifica la ubicación del bloque dentro de la cadena.
- `tableId`: el identificador único de la tabla cuya revisión está editando.
- `documentId`: el identificador único del documento de la revisión que se va a editar.
- `version`: el número de versión del documento de la revisión que se va a editar.

A continuación se muestra un ejemplo de estructura de retorno con datos de ejemplo.

```
{
  blockAddress: {
    strandId: "CsRnx0RDoNK6ANEEePa1ov",
    sequenceNo: 134
  },
  tableId: "6GZumdHggklLdMGyQq9DNX",
  documentId: "IXlQPSbfyKMIIsygePeKrZ",
  version: 0
}
```

Ejemplos

```
EXEC REDACT_REVISION `{strandId:"7z2P0AyQKWD8oFYmGNhi8D", sequenceNo:7}`,
 '8F0TPCmdNQ6JTRpiLj2TmW', '05K8zpGYWynDlEOK5afDRc'
```

Operadores PartiQL en Amazon QLDB

PartiQL en Amazon QLDB admite los siguientes operadores estándar de SQL.

> **Note**
>
> QLDB no admite actualmente ningún operador SQL que no se incluya en esta lista.

Operadores aritméticos

Operador	Descripción
+	Add
-	Subtract (Sustracción)
*	Multiply (Multiplicación)
/	Divide (División)
%	Módulo

Operadores de comparación

Operador	Descripción
=	Igual que
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	No igual que

Logical operators (Operadores lógicos)

Operador	Descripción
`AND`	`TRUE` si todas las condiciones separadas por `AND` son `TRUE`
`BETWEEN`	`TRUE` si el operado se inscribe en el rango de comparaciones
`IN`	`TRUE` si el operando es igual a uno de una lista de expresiones
`IS`	`TRUE` si el operando es un tipo de datos dado, incluyendo `NULL` o `MISSING`
`LIKE`	`TRUE` si el operando coincide con un patrón
`NOT`	Invierte el valor de una expresión booleana dada
`OR`	`TRUE` si alguna de las condiciones está separada por `OR` es `TRUE`

Operadores de cadena

Operador	Descripción
`||`	Concatena dos cadenas a ambos extremos del operador `||` y devuelve una cadena concatena da. Si una o ambas cadenas son `NULL`, el resultado de la concatenación es null.

Palabras clave reservadas en Amazon QLDB

A continuación, se ofrece una lista de las palabras clave de PartiQL reservadas en Amazon QLDB. Puede utilizar una palabra clave reservada como identificador entre comillas dobles (por ejemplo, `"user"`). Para obtener información sobre las convenciones para comillas de PartiQL en QLDB, consulte Consulta de Ion con PartiQL.

> **Important**
>
> Todas las palabras clave de esta lista se consideran reservadas porque PartiQL es compatible con versiones anteriores de SQL-92. Sin embargo, QLDB solo admite un subconjunto de estas palabras reservadas. Para ver la lista de las palabras clave SQL que QLDB admite actualmente, consulte los siguientes temas:
>
> - Funciones de PartiQL
> - Operadores PartiQL
> - Comandos de PartiQL

```
ABSOLUTE
ACTION
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BAG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BLOB
```

```
BOOL
BOOLEAN
BOTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHARACTER
CHARACTER_LENGTH
CHAR_LENGTH
CHECK
CLOB
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATE
DATE_ADD
DATE_DIFF
DAY
DEALLOCATE
DEC
DECIMAL
```

```
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END
END-EXEC
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR
FOREIGN
FOUND
FROM
FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IMMEDIATE
```

```
IN
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEADING
LEFT
LEVEL
LIKE
LIMIT
LIST
LOCAL
LOWER
MATCH
MAX
MIN
MINUTE
MISSING
MODULE
MONTH
NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NO
NOT
NULL
NULLIF
NUMERIC
```

```
OCTET_LENGTH
OF
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PAD
PARTIAL
PIVOT
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
READ
REAL
REFERENCES
RELATIVE
REMOVE
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SESSION
SESSION_USER
SET
SEXP
SIZE
SMALLINT
```

```
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLSTATE
STRING
STRUCT
SUBSTRING
SUM
SYMBOL
SYSTEM_USER
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TO_STRING
TO_TIMESTAMP
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIM
TRUE
TUPLE
TXID
UNDROP
UNION
UNIQUE
UNKNOWN
UNPIVOT
UPDATE
UPPER
USAGE
USER
USING
UTCNOW
VALUE
VALUES
VARCHAR
```

```
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE
```

Referencia del formato de datos de Amazon Ion en Amazon QLDB

Amazon QLDB utiliza un modelo de notación de datos que unifica Amazon Ion con un subconjunto de tipos de PartiQL. Esta sección proporciona una descripción general de referencia del formato de datos del documento Ion, independiente de su integración con PartiQL.

Consulta de Ion con PartiQL en Amazon QLDB

Para obtener información sobre la sintaxis y la semántica de la consulta de datos de Ion con PartiQL en QLDB, consulte Consulta de Ion con PartiQL en la referencia de PartiQL de Amazon QLDB.

Para ver ejemplos de código que consultan y procesan datos de Ion en un libro mayor de QLDB, consulte Ejemplos de código de Amazon Ion y Trabajar con Amazon Ion.

Temas

- ¿Qué es Amazon Ion?
- Especificación de Ion
- Compatible con JSON
- Extensiones de JSON
- Ejemplo de texto de Ion
- Referencias de API
- Ejemplos de código de Amazon Ion en QLDB

¿Qué es Amazon Ion?

Ion es un formato de serialización de datos jerárquicos, autodescriptivo, altamente tipificado y de código abierto que se desarrolló originalmente de manera interna en Amazon. Se basa en un

modelo de datos abstracto que permite almacenar datos estructurados y no estructurados. Es un superconjunto de JSON, lo que significa que cualquier documento JSON válido también es un documento Ion válido. En esta guía se parte de un conocimiento de usuario básico de JSON. Si aún no está familiarizado con JSON, consulte Introducción a JSON para obtener más información.

Puede realizar las notaciones de los documentos de Ion de forma intercambiable, ya sea en forma de texto legible por seres humanos o en formato codificado en binario. Al igual que JSON, el formato de texto es fácil de leer y escribir, y permite la creación rápida de prototipos. La codificación binaria es más compacta y eficiente para persistir, transmitir y analizar. Un procesador Ion puede transcodificar entre ambos formatos para representar exactamente el mismo conjunto de estructuras de datos sin pérdida de datos. Esta característica permite a las aplicaciones optimizar la forma en que procesan los datos para distintos casos de uso.

> **Note**
>
> El modelo de datos de Ion se basa estrictamente en valores y no admite referencias. Por lo tanto, el modelo de datos puede representar jerarquías de datos que se pueden anidar con una profundidad arbitraria, pero no gráficos dirigidos.

Especificación de Ion

Consulte el documento de especificaciones de Ion en el sitio de Amazon GitHub para obtener una lista completa de los tipos de datos principales de Ion con descripciones completas y detalles de formato de valores.

Para agilizar el desarrollo de aplicaciones, Amazon Ion proporciona bibliotecas cliente que procesan los datos de Ion por usted. Para ver ejemplos de código de casos de uso frecuente para procesar datos de Ion, consulte el Libro de recetas de Amazon Ion en GitHub.

Compatible con JSON

Al igual que con JSON, los documentos de Amazon Ion se componen de un conjunto de tipos de datos primitivos y un conjunto de tipos de contenedores definidos de forma recursiva. Ion incluye los siguientes tipos de datos JSON tradicionales:

- `null`: un valor null (vacío) genérico y sin tipo. Además, como se describe en la siguiente sección, Ion admite un tipo null distinto para cada tipo primitivo.

- `bool`: valores booleanos.
- `string`: literales de texto Unicode.
- `list`: colecciones de valores heterogéneas ordenadas.
- `struct`: colecciones desordenadas de pares nombre-valor Al igual que JSON, `struct` permite varios valores por nombre, pero generalmente no se recomienda hacerlo.

Extensiones de JSON

Tipos de número

En lugar del tipo `number` JSON ambiguo, Amazon Ion define estrictamente los números como uno de los siguientes tipos:

- `int`: enteros firmados de tamaño arbitrario.
- `decimal`: números reales codificados con decimales de precisión arbitraria.
- `float`: números de coma flotante codificados en binario (IEEE de 64 bits).

Al analizar documentos, un procesador Ion asigna los siguientes tipos de números:

- `int`: números sin exponente ni coma decimal (por ejemplo, `100200`).
- `decimal`: números con coma decimal sin exponente (por ejemplo, `0.00001`, `200.0`).
- `float`: números con exponente, como la notación científica o la notación electrónica (por ejemplo, `2e0`, `3.1e-4`).

Nuevos tipos de datos

Amazon Ion añade los siguientes tipos de datos:

- `timestamp`: momentos de fecha, hora y zona horaria de precisión arbitraria.
- `symbol`: átomos simbólicos de Unicode (como los identificadores).
- `blob`: datos binarios de codificación definida por el usuario.
- `clob`: datos de texto de codificación definida por el usuario.
- `sexp`: colecciones ordenadas de valores con semántica definida por la aplicación.

Tipos null

Además del tipo null genérico definido por JSON, Amazon Ion admite un tipo null distinto para cada tipo primitivo. Esto indica una falta de valor y, al mismo tiempo, mantiene un tipo de datos estricto.

```
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
```

Ejemplo de texto de Ion

```
// Here is a struct, which is similar to a JSON object.
{
    // Field names don't always have to be quoted.
    name: "fido",

    // This is an integer.
    age: 7,

    // This is a timestamp with day precision.
    birthday: 2012-03-01T,

    // Here is a list, which is like a JSON array.
    toys: [
        // These are symbol values, which are like strings,
        // but get encoded as integers in binary.
        ball,
        rope
    ],
}
```

Referencias de API

- ion-go
- ion-java
- ion-js
- ion-python

Ejemplos de código de Amazon Ion en QLDB

En esta sección se proporcionan ejemplos de código que procesan los datos de Amazon Ion leyendo y escribiendo valores de documentos en un libro mayor de Amazon QLDB. Los ejemplos de código utilizan el controlador de QLDB para ejecutar instrucciones de PartiQL en el libro mayor. Estos ejemplos forman parte de la aplicación de ejemplo en Tutorial de introducción a Amazon QLDB mediante un ejemplo de aplicación y son de código abierto en el sitio GitHub de muestras de AWS.

Para ver ejemplos de código de casos de uso frecuente para procesar datos de Ion, consulte el Libro de recetas de Amazon Ion en GitHub.

Ejecutar el código

El código del tutorial de cada lenguaje de programación comprende los siguientes pasos:

1. Conéctese al libro mayor de muestras `vehicle-registration`.
2. Cree una tabla denominada `IonTypes`.
3. Inserte un documento en la tabla con un solo campo `Name`.
4. Para cada tipo de datos de Ion compatible:
   a. Actualice el campo `Name` del documento con un valor literal del tipo de datos.
   b. Consulte la tabla para obtener la revisión más reciente del documento.
   c. Asegúrese de que el valor de `Name` conserva sus propiedades de tipo de datos originales comprobando que coincide con el tipo esperado.
5. Elimine la tabla `IonTypes`.

> **Note**
>
> Antes de ejecutar este código de tutorial, debe crear un libro mayor llamado `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()
```

Referencia de la API de Amazon QLDB

En este capítulo se describen las operaciones de API de bajo nivel para Amazon QLDB a las que se puede acceder mediante HTTP, la AWS Command Line Interface (AWS CLI) o un SDK de AWS:

- Amazon QLDB: la API de administración de recursos de QLDB (también conocida como plano de control). Esta API se usa únicamente para administrar los recursos del libro mayor y para las operaciones de datos no transaccionales. Puede usar estas operaciones para crear, eliminar, describir, enumerar y actualizar libros mayores. También puede verificar los datos criptográficamente y exportar o transmitir bloques de secuencia.
- Sesión de Amazon QLDB: la API de datos transaccionales de QLDB. Puede usar esta API para ejecutar transacciones de datos en un libro mayor con instrucciones PartiQL.

> **Important**
>
> En lugar de interactuar directamente con esta API de sesión de QLDB, recomendamos usar el controlador QLDB o el intérprete de comandos QLDB para ejecutar transacciones de datos en un libro mayor.
>
> - Si trabaja con un SDK de AWS, use el controlador QLDB. El controlador proporciona una capa de abstracción de alto nivel sobre la API de datos de sesión de QLDB, y gestiona la operación `SendCommand` por usted. Para obtener más información y ver una lista de los lenguajes de programación compatibles, consulte Introducción al controlador.
> - Si está trabajando con la AWS CLI, use el intérprete de comandos QLDB. El intérprete de comandos es una interfaz de línea de comandos que usa el controlador QLDB para interactuar con un libro mayor. Para obtener información, consulte Uso del intérprete de comandos de Amazon QLDB (solo API de datos).

Temas

Acciones

Amazon QLDB admite las siguientes acciones:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

La sesión de Amazon QLDB admite las siguientes acciones:

- SendCommand

Amazon QLDB

Amazon QLDB admite las siguientes acciones:

- CancelJournalKinesisStream
- CreateLedger
- DeleteLedger
- DescribeJournalKinesisStream
- DescribeJournalS3Export
- DescribeLedger
- ExportJournalToS3
- GetBlock
- GetDigest
- GetRevision
- ListJournalKinesisStreamsForLedger
- ListJournalS3Exports
- ListJournalS3ExportsForLedger
- ListLedgers
- ListTagsForResource
- StreamJournalToKinesis
- TagResource
- UntagResource
- UpdateLedger
- UpdateLedgerPermissionsMode

CancelJournalKinesisStream

Servicio: Amazon QLDB

Finaliza una secuencia del diario de Amazon QLDB determinada. Para poder cancelar una secuencia, su estado actual debe ser `ACTIVE`.

No puede reiniciar una secuencia después de cancelarla. Los recursos de secuencia de QLDB cancelados están sujetos a un período de retención de 7 días, por lo que se eliminan automáticamente una vez vencido este límite.

Sintaxis de la solicitud

```
DELETE /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

streamId

El UUID (representado en texto codificado en Base62) de la secuencia del diario QLDB que va a cancelar.

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "StreamId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

StreamId

El UUID (texto codificado en Base62) de la secuencia del diario QLDB cancelada.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

CreateLedger

Servicio: Amazon QLDB

Crea un nuevo libro mayor Cuenta de AWS en tu región actual.

Sintaxis de la solicitud

```
POST /ledgers HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "Tags": {
      "string" : "string"
   }
}
```

Parámetros de solicitud del URI

La solicitud no utiliza ningún parámetro de URI.

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

DeletionProtection

Especifica si el libro mayor está protegido contra la eliminación por parte de cualquier usuario. Si no se define al crear el libro mayor, esta característica estará habilitada (`true`) de forma predeterminada.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede llevar a cabo la desactivación mediante una llamada a la operación `UpdateLedger` para establecer este parámetro en `false`.

Tipo: Booleano

Obligatorio: no

KmsKey

La clave in AWS Key Management Service (AWS KMS) que se utilizará para cifrar los datos almacenados en el libro mayor. Para obtener más información, consulte Encryption at rest (Cifrado en reposo) en la Guía para desarrolladores de Amazon QLDB.

Utilice una de las siguientes opciones para especificar este parámetro:

- `AWS_OWNED_KMS_KEY`: utilice una AWS KMS clave que sea de su propiedad y que esté AWS gestionada por usted.
- Indefinido: de forma predeterminada, usa una clave de AWS KMS propia.
- Una clave KMS válida y simétrica que administra el cliente: utilice la clave KMS de cifrado simétrico especificada en la cuenta que ha creado y que posee y administra.

  Amazon QLDB no es compatible con claves asimétricas. Para obtener más información, consulte Uso de claves simétricas y asimétricas en la AWS Key Management Service Guía para desarrolladores.

Para especificar una clave KMS que administre el cliente, utilice el ID de clave, el nombre de recurso de Amazon (ARN), el nombre de alias o bien el ARN de alias. Cuando utilice un nombre de alias, utilice el prefijo `"alias/"`. Para especificar una clave en otra Cuenta de AWS, debe usar la clave ARN o el alias ARN.

Por ejemplo:

- ID de clave: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN de clave: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nombre de alias: `alias/ExampleAlias`
- ARN de alias: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Para obtener más información, consulte los identificadores clave (KeyId) en la AWS Key Management Service Guía para desarrolladores.

Tipo: cadena

Limitaciones de longitud: longitud máxima de 1600 caracteres.

Obligatorio: no

Name

El nombre del libro mayor que desea crear. El nombre debe ser único entre todos los libros de contabilidad de la Cuenta de AWS región actual.

Las restricciones en la nomenclatura para los nombres de contabilidad se definen en Cuotas en Amazon QLDB en la Guía para desarrolladores de Amazon QLDB.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

PermissionsMode

El modo de permisos que se va a asignar al libro mayor que desea crear. Este parámetro puede tener uno de los siguientes valores:

- `ALLOW_ALL`: un modo de permisos heredado que habilita el control de acceso con granularidad de la API para los libros mayores.

  Este modo permite a los usuarios que tengan el permiso de API `SendCommand` para este libro mayor poner en marcha todos los comandos de PartiQL (por lo tanto, `ALLOW_ALL`) en cualquier tabla del libro mayor especificado. Este modo no tendrá en cuenta las políticas de permisos de IAM de la tabla o comando que cree para el libro mayor.

- `STANDARD`: (recomendado) un modo de permisos que habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL.

  De forma predeterminada, este modo deniega todas las solicitudes de los usuarios para poner en marcha cualquier comando de PartiQL en tablas de este libro mayor. Para permitir que se pongan en marcha comandos de PartiQL, debe crear políticas de permisos de IAM para recursos de tablas y acciones de PartiQL específicos, además del permiso de API `SendCommand` para el libro mayor. Para obtener más información, consulte Getting started with the standard permissions mode (Introducción al modo de permisos estándar) en la Guía para desarrolladores de Amazon QLDB.

> **Note**
>
> Recomendamos encarecidamente el modo `STANDARD` de permisos para maximizar la seguridad de los datos del libro mayor.

Tipo: cadena

Valores válidos: `ALLOW_ALL | STANDARD`

Obligatorio: sí

Tags

Los pares clave-valor que quiera agregar como etiquetas a al libro mayor que desea crear. Las claves de etiqueta distinguen entre mayúsculas y minúsculas. Los valores de etiquetas distinguen entre mayúsculas y minúsculas y pueden ser nulos.

Tipo: mapa de cadena a cadena

Entradas de mapa: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Limitaciones de longitud de la clave: longitud mínima de 1. Longitud máxima de 128.

Limitaciones de longitud de los valores: longitud mínima de 0. La longitud máxima es de 256 caracteres.

Obligatorio: no

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "KmsKeyArn": "string",
   "Name": "string",
   "PermissionsMode": "string",
   "State": "string"
```

```
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Arn

El nombre de recurso de Amazon (ARN) para el libro mayor.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

CreationDateTime

La fecha y la hora, en formato de tiempo epoch, en que se creó el libro mayor. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Tipo: marca temporal

DeletionProtection

Especifica si el libro mayor está protegido contra la eliminación por parte de cualquier usuario. Si no se define al crear el libro mayor, esta característica estará habilitada (`true`) de forma predeterminada.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede llevar a cabo la desactivación mediante una llamada a la operación `UpdateLedger` para establecer este parámetro en `false`.

Tipo: Booleano

KmsKeyArn

El ARN de la clave KMS administrada por el cliente que emplea el libro mayor para el cifrado en reposo. Si este parámetro no está definido, el libro mayor utiliza una clave KMS propia AWS para el cifrado.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Name

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

El modo de permisos del libro mayor que ha creado.

Tipo: cadena

Valores válidos: `ALLOW_ALL | STANDARD`

State

El estado actual del libro mayor.

Tipo: cadena

Valores válidos: `CREATING | ACTIVE | DELETING | DELETED`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

LimitExceededException

Ha alcanzado el límite del número máximo de recursos permitidos.

Código de estado HTTP: 400

ResourceAlreadyExistsException

El recurso especificado ya existe.

Código de estado HTTP: 409

ResourceInUseException

El recurso especificado no se puede modificar en este momento.

Código de estado HTTP: 409

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DeleteLedger

Servicio: Amazon QLDB

Elimina el libro mayor y todo su contenido. Esta acción es irreversible.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede llevar a cabo la desactivación mediante una llamada a la operación `UpdateLedger` para establecer este parámetro en `false`.

Sintaxis de la solicitud

```
DELETE /ledgers/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del libro mayor que desea eliminar.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado no se puede modificar en este momento.

Código de estado HTTP: 409

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

DescribeJournalKinesisStream

Servicio: Amazon QLDB

Devuelve información detallada sobre una secuencia del diario determinada de Amazon QLDB. El resultado incluye el Nombre de recurso de Amazon (ARN), el nombre de la secuencia, el estado actual, la hora de creación y los parámetros de la solicitud de creación de la secuencia original.

Esta acción no devuelve ninguna secuencia del diario caducada. Para obtener más información, consulte caducidad de secuencias de terminal en la Guía para desarrolladores de Amazon QLDB.

Sintaxis de la solicitud

```
GET /ledgers/name/journal-kinesis-streams/streamId HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

streamId

El UUID (representado en texto codificado en Base62) de la secuencia del diario QLDB que va a describir.

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Stream": {
      "Arn": "string",
      "CreationTime": number,
      "ErrorCause": "string",
      "ExclusiveEndTime": number,
      "InclusiveStartTime": number,
      "KinesisConfiguration": {
         "AggregationEnabled": boolean,
         "StreamArn": "string"
      },
      "LedgerName": "string",
      "RoleArn": "string",
      "Status": "string",
      "StreamId": "string",
      "StreamName": "string"
   }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Stream

Información sobre la secuencia del diario QLDB devuelta por una solicitud `DescribeJournalS3Export`.

Tipo: objeto JournalKinesisStreamDescription

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DescribeJournalS3Export

Servicio: Amazon QLDB

Información sobre un trabajo de exportación de diario que incluye el nombre del libro mayor, el ID de exportación, la hora de creación, el estado actual y los parámetros de la solicitud de creación de exportación original.

Esta acción no devuelve ningún trabajo de exportación caducado. Para obtener más información, consulte caducidad de trabajos de exportación en la Guía para desarrolladores de Amazon QLDB.

Si el trabajo de exportación con el `ExportId` dado no existe, devuelve `ResourceNotFoundException`.

Si el libro mayor con el `Name` dado no existe, devuelve `ResourceNotFoundException`.

Sintaxis de la solicitud

```
GET /ledgers/name/journal-s3-exports/exportId HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

exportId

El UUID (representado en texto codificado en Base62) del trabajo de exportación del diario que se va a describir.

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportDescription": {
      "ExclusiveEndTime": number,
      "ExportCreationTime": number,
      "ExportId": "string",
      "InclusiveStartTime": number,
      "LedgerName": "string",
      "OutputFormat": "string",
      "RoleArn": "string",
      "S3ExportConfiguration": {
         "Bucket": "string",
         "EncryptionConfiguration": {
            "KmsKeyArn": "string",
            "ObjectEncryptionType": "string"
         },
         "Prefix": "string"
      },
      "Status": "string"
   }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

ExportDescription

Información sobre el trabajo de exportación de diarios devuelto por una solicitud `DescribeJournalS3Export`.

Tipo: objeto JournalS3ExportDescription

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

DescribeLedger

Servicio: Amazon QLDB

Devuelve información sobre un libro mayor, incluyendo su estado, modo de permisos, configuración de cifrado en reposo y cuándo se creó.

Sintaxis de la solicitud

```
GET /ledgers/name HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre del libro mayor que desea describir.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "EncryptionDescription": {
      "EncryptionStatus": "string",
      "InaccessibleKmsKeyDateTime": number,
      "KmsKeyArn": "string"
   },
```

```
    "Name": "string",
    "PermissionsMode": "string",
    "State": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Arn

El nombre de recurso de Amazon (ARN) para el libro mayor.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

CreationDateTime

La fecha y la hora, en formato de tiempo epoch, en que se creó el libro mayor. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Tipo: marca temporal

DeletionProtection

Especifica si el libro mayor está protegido contra la eliminación por parte de cualquier usuario. Si no se define al crear el libro mayor, esta característica estará habilitada (`true`) de forma predeterminada.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede llevar a cabo la desactivación mediante una llamada a la operación `UpdateLedger` para establecer este parámetro en `false`.

Tipo: Booleano

EncryptionDescription

Información sobre el cifrado de los datos en reposo del libro mayor. Esto incluye el estado actual, la AWS KMS clave y cuándo se volvió inaccesible a la clave (en caso de error). Si este parámetro no está definido, el libro mayor utiliza una clave KMS propia AWS para el cifrado.

Tipo: objeto LedgerEncryptionDescription

Name

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

El modo de permisos del libro mayor.

Tipo: cadena

Valores válidos: `ALLOW_ALL | STANDARD`

State

El estado actual del libro mayor.

Tipo: cadena

Valores válidos: `CREATING | ACTIVE | DELETING | DELETED`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ExportJournalToS3

Servicio: Amazon QLDB

Exporta el contenido del diario en un intervalo de fechas y horas de un libro mayor a un bucket de Amazon Simple Storage Service (Amazon S3) especificado. Un trabajo de exportación del diario puede escribir los objetos de datos en el formato de texto o binario de Amazon Ion, o en el formato de texto JSON Lines.

Si el libro mayor con el `Name` dado no existe, devuelve `ResourceNotFoundException`.

Si el estado del libro mayor con el `Name` dado es `CREATING`, devuelve `ResourcePreconditionNotMetException`.

Puede iniciar hasta dos solicitudes de exportación de diarios simultáneas para cada libro mayor. Más allá de este límite, las solicitudes de exportación del diario devuelven `LimitExceededException`.

Sintaxis de la solicitud

```
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"
   }
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

ExclusiveEndTime

La fecha y hora de finalización exclusivas del rango de contenidos del diario que desea exportar.

La `ExclusiveEndTime` debe estar en formato de fecha y hora `ISO 8601` y en hora universal coordinada (UTC). Por ejemplo: `2019-06-13T21:36:34Z`.

La `ExclusiveEndTime` debe ser menor o igual que la fecha y hora UTC actuales.

Tipo: marca temporal

Obligatorio: sí

InclusiveStartTime

La fecha y hora de inicio inclusivas del rango de contenidos del diario que desea exportar.

La `InclusiveStartTime` debe estar en formato de fecha y hora `ISO 8601` y en hora universal coordinada (UTC). Por ejemplo: `2019-06-13T21:36:34Z`.

La `InclusiveStartTime` debe ser anterior a `ExclusiveEndTime`.

Si proporciona una `InclusiveStartTime` anterior a la `CreationDateTime` del libro mayor, Amazon QLDB la asigna de manera predeterminada a la `CreationDateTime` del libro mayor.

Tipo: marca temporal

Obligatorio: sí

OutputFormat

El formato de salida de los datos de su diario exportado. Un trabajo de exportación del diario puede escribir los objetos de datos en el formato de texto o binario de Amazon Ion, o en el formato de texto JSON Lines.

Valor predeterminado: `ION_TEXT`

En el formato de JSON Lines, cada bloque del diario de un objeto de datos exportado es un objeto JSON válido que está delimitado por saltos de línea. Puede usar este formato para integrar directamente las exportaciones de JSON con herramientas de análisis como Amazon Athena y AWS Glue porque estos servicios pueden analizar automáticamente el JSON delimitado por saltos de línea.

Tipo: cadena

Valores válidos: `ION_BINARY | ION_TEXT | JSON`

Obligatorio: no

RoleArn

El nombre de recurso de Amazon (ARN) del rol de IAM que concede permisos a QLDB para que los trabajos de exportación de diarios hagan lo siguiente:

- Escribe objetos en un bucket de Amazon S3.
- (Opcional) Utilice su clave gestionada por el cliente AWS Key Management Service (AWS KMS) para cifrar los datos exportados en el servidor.

Para transferir un rol a QLDB al solicitar una exportación de diario, debe tener permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM. Esto es necesario para todas las solicitudes de exportación de diarios.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

S3ExportConfiguration

Los ajustes de la configuración del bucket de Amazon S3 de destino de su solicitud de exportación.

Tipo: objeto S3ExportConfiguration

Obligatorio: sí

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "ExportId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

ExportId

El UUID (representado en texto codificado en Base62) que QLDB asigna a cada trabajo de exportación de diario.

Para describir su solicitud de exportación y comprobar el estado del trabajo, puede utilizar `ExportId` para llamar a `DescribeJournalS3Export`.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetBlock

Servicio: Amazon QLDB

Devuelve un objeto de bloque en una dirección específica de un diario. También devuelve una prueba del bloque especificado para su verificación, si `DigestTipAddress` se proporciona.

Para obtener información sobre el contenido de datos de un bloque, consulte el contenido del diario en la Guía para desarrolladores de Amazon QLDB.

Si el libro mayor especificado no existe o está en estado `DELETING`, devuelve `ResourceNotFoundException`.

Si el libro mayor especificado está en estado `CREATING`, devuelve `ResourcePreconditionNotMetException`.

Si no existe ningún bloque con la dirección especificada, devuelve `InvalidParameterException`.

Sintaxis de la solicitud

```
POST /ledgers/name/block HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

BlockAddress

La ubicación del bloque que quiere solicitar. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Por ejemplo: `{strandId:"BlFTjlSXze9BIh1K0szcE3",sequenceNo:14}`.

Tipo: objeto ValueHolder

Obligatorio: sí

DigestTipAddress

La última ubicación de bloque incluida en el resumen para la que se solicita una prueba. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Por ejemplo: `{strandId:"BlFTjlSXze9BIh1K0szcE3",sequenceNo:49}`.

Tipo: objeto ValueHolder

Obligatorio: no

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Block": {
      "IonText": "string"
   },
   "Proof": {
      "IonText": "string"
   }
```

```
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Block

El objeto de datos de bloque en formato de Amazon Ion.

Tipo: objeto ValueHolder

Proof

El objeto de prueba en formato de Amazon Ion devuelto por una solicitud `GetBlock`. Una prueba contiene la lista de valores hash necesarios para volver a calcular el resumen especificado mediante un árbol de Merkle, empezando por el bloque especificado.

Tipo: objeto ValueHolder

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetDigest

Servicio: Amazon QLDB

Devuelve el resumen de un libro mayor en el último bloque confirmado del diario. La respuesta incluye un valor hash de 256 bits y una dirección de bloque.

Sintaxis de la solicitud

```
POST /ledgers/name/digest HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Digest": blob,
   "DigestTipAddress": {
      "IonText": "string"
   }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Digest

El valor de hash de 256 bits que representa el resumen devuelto por una solicitud `GetDigest`.

Tipo: objeto de datos binarios codificados en Base64

Limitaciones de longitud: longitud fija de 32.

DigestTipAddress

La última ubicación de bloque incluida en el resumen que ha solicitado. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Tipo: objeto ValueHolder

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

GetRevision

Servicio: Amazon QLDB

Devuelve un objeto de datos de revisión para un identificador de documento y una dirección de bloque específicos. También devuelve una prueba de la revisión especificada para su verificación, si se proporciona `DigestTipAddress`.

Sintaxis de la solicitud

```
POST /ledgers/name/revision HTTP/1.1
Content-type: application/json

{
   "BlockAddress": {
      "IonText": "string"
   },
   "DigestTipAddress": {
      "IonText": "string"
   },
   "DocumentId": "string"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

BlockAddress

La ubicación en el bloque de la revisión del documento que se va a verificar. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Por ejemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:14}`.

Tipo: objeto ValueHolder

Obligatorio: sí

DigestTipAddress

La última ubicación de bloque incluida en el resumen para la que se solicita una prueba. La dirección es una estructura de Amazon Ion que consta de dos campos: `strandId` y `sequenceNo`.

Por ejemplo: `{strandId:"BlFTjlSXze9BIh1KOszcE3",sequenceNo:49}`.

Tipo: objeto ValueHolder

Obligatorio: no

DocumentId

El UUID (representado en texto codificado en Base62) del documento que se va a verificar.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Proof": {
      "IonText": "string"
```

```
    },
    "Revision": {
       "IonText": "string"
    }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Proof

El objeto de prueba en formato de Amazon Ion devuelto por una solicitud `GetRevision`. Una prueba contiene la lista de valores hash necesarios para volver a calcular el resumen especificado mediante un árbol de Merkle, empezando por la revisión del documento especificada.

Tipo: objeto ValueHolder

Revision

El objeto de datos de revisión del documento en formato de Amazon Ion.

Tipo: objeto ValueHolder

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListJournalKinesisStreamsForLedger

Servicio: Amazon QLDB

Crea una secuencia de diario de Amazon QLDB para un libro mayor determinado.

Esta acción no devuelve ninguna secuencia del diario caducada. Para obtener más información, consulte caducidad de secuencias de terminal en la Guía para desarrolladores de Amazon QLDB.

Esta acción devuelve un máximo de objetos `MaxResults`. Está paginada para que pueda recuperar todos los elementos llamando a `ListJournalKinesisStreamsForLedger` varias veces.

Sintaxis de la solicitud

```
GET /ledgers/name/journal-kinesis-streams?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

MaxResults

El número máximo de resultados que se devuelven en una única solicitud `ListJournalKinesisStreamsForLedger`. (Es posible que el número real de resultados devueltos sea inferior).

Rango válido: valor mínimo de 1. Valor máximo de 100.

NextToken

Un token de paginación que indica que desea recuperar la siguiente página de resultados. Si recibió un valor para `NextToken` en la respuesta de una llamada anterior a `ListJournalKinesisStreamsForLedger`, debe usar ese valor como entrada aquí.

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Streams": [
      {
         "Arn": "string",
         "CreationTime": number,
         "ErrorCause": "string",
         "ExclusiveEndTime": number,
         "InclusiveStartTime": number,
         "KinesisConfiguration": {
            "AggregationEnabled": boolean,
            "StreamArn": "string"
         },
         "LedgerName": "string",
         "RoleArn": "string",
         "Status": "string",
         "StreamId": "string",
         "StreamName": "string"
      }
   ]
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

NextToken

- Si `NextToken` está vacío, se ha procesado la última página de resultados y no hay más resultados que recuperar.

- Si `NextToken` no está vacío, hay más resultados disponibles. Para recuperar la siguiente página de resultados, use el valor de `NextToken` en una llamada posterior a `ListJournalKinesisStreamsForLedger`.

  Tipo: cadena

  Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

  Patrón: `^[A-Za-z-0-9+/=]+$`

Streams

Las secuencias del diario de QLDB que están asociadas actualmente al libro mayor determinado.

Tipo: matriz de objetos JournalKinesisStreamDescription

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListJournalS3Exports

Servicio: Amazon QLDB

Devuelve todos los trabajos de exportación de diarios de todos los libros mayores asociados a la Cuenta de AWS y la región actuales.

Esta acción devuelve un máximo de elementos `MaxResults` y está paginada para que pueda recuperar todos los elementos llamando varias veces a `ListJournalS3Exports`.

Esta acción no devuelve ningún trabajo de exportación caducado. Para obtener más información, consulte caducidad de trabajos de exportación en la Guía para desarrolladores de Amazon QLDB.

Sintaxis de la solicitud

```
GET /journal-s3-exports?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

MaxResults

El número máximo de resultados que se devuelven en una única solicitud `ListJournalS3Exports`. (Es posible que el número real de resultados devueltos sea inferior).

Rango válido: valor mínimo de 1. Valor máximo de 100.

NextToken

Un token de paginación que indica que desea recuperar la siguiente página de resultados. Si recibió un valor para `NextToken` en la respuesta de una llamada `ListJournalS3Exports` anterior, debe usar ese valor como entrada aquí.

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "JournalS3Exports": [
      {
         "ExclusiveEndTime": number,
         "ExportCreationTime": number,
         "ExportId": "string",
         "InclusiveStartTime": number,
         "LedgerName": "string",
         "OutputFormat": "string",
         "RoleArn": "string",
         "S3ExportConfiguration": {
            "Bucket": "string",
            "EncryptionConfiguration": {
               "KmsKeyArn": "string",
               "ObjectEncryptionType": "string"
            },
            "Prefix": "string"
         },
         "Status": "string"
      }
   ],
   "NextToken": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

JournalS3Exports

Los trabajos de exportación de diarios de todos los libros mayores asociados a la Cuenta de AWS y la región actuales.

Tipo: matriz de objetos JournalS3ExportDescription

NextToken

- Si `NextToken` está vacío, se ha procesado la última página de resultados y no hay más resultados que recuperar.
- Si `NextToken` no está vacío, hay más resultados disponibles. Para recuperar la siguiente página de resultados, use el valor de `NextToken` en una llamada posterior a `ListJournalS3Exports`.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListJournalS3ExportsForLedger

Servicio: Amazon QLDB

Devuelve todos los trabajos de exportación del diario de un libro mayor especificado.

Esta acción devuelve un máximo de elementos `MaxResults` y está paginada para que pueda recuperar todos los elementos llamando varias veces a `ListJournalS3ExportsForLedger`.

Esta acción no devuelve ningún trabajo de exportación caducado. Para obtener más información, consulte caducidad de trabajos de exportación en la Guía para desarrolladores de Amazon QLDB.

Sintaxis de la solicitud

```
GET /ledgers/name/journal-s3-exports?max_results=MaxResults&next_token=NextToken
 HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

MaxResults

El número máximo de resultados que se devuelven en una única solicitud `ListJournalS3ExportsForLedger`. (Es posible que el número real de resultados devueltos sea inferior).

Rango válido: valor mínimo de 1. Valor máximo de 100.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

NextToken

Un token de paginación que indica que desea recuperar la siguiente página de resultados. Si recibió un valor para `NextToken` en la respuesta de una llamada `ListJournalS3ExportsForLedger` anterior, debe usar ese valor como entrada aquí.

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "JournalS3Exports": [
      {
         "ExclusiveEndTime": number,
         "ExportCreationTime": number,
         "ExportId": "string",
         "InclusiveStartTime": number,
         "LedgerName": "string",
         "OutputFormat": "string",
         "RoleArn": "string",
         "S3ExportConfiguration": {
            "Bucket": "string",
            "EncryptionConfiguration": {
               "KmsKeyArn": "string",
               "ObjectEncryptionType": "string"
            },
            "Prefix": "string"
         },
         "Status": "string"
      }
   ],
   "NextToken": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

JournalS3Exports

Los trabajos de exportación de diarios que están asociados actualmente al libro mayor especificado.

Tipo: matriz de objetos JournalS3ExportDescription

NextToken

- Si `NextToken` está vacío, se ha procesado la última página de resultados y no hay más resultados que recuperar.
- Si `NextToken` no está vacío, hay más resultados disponibles. Para recuperar la siguiente página de resultados, use el valor de `NextToken` en una llamada posterior a `ListJournalS3ExportsForLedger`.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python

- AWS SDK para Ruby V3

ListLedgers

Servicio: Amazon QLDB

Devuelve todos los libros de contabilidad asociados a la corriente Cuenta de AWS y a la región.

Esta acción devuelve un máximo de elementos `MaxResults` y está paginada para que pueda recuperar todos los elementos llamando varias veces a `ListLedgers`.

Sintaxis de la solicitud

```
GET /ledgers?max_results=MaxResults&next_token=NextToken HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

MaxResults

El número máximo de resultados que se devuelven en una única solicitud `ListLedgers`. (Es posible que el número real de resultados devueltos sea inferior).

Rango válido: valor mínimo de 1. Valor máximo de 100.

NextToken

Un token de paginación que indica que desea recuperar la siguiente página de resultados. Si recibió un valor para `NextToken` en la respuesta de una llamada `ListLedgers` anterior, debe usar ese valor como entrada aquí.

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Ledgers": [
      {
         "CreationDateTime": number,
         "Name": "string",
         "State": "string"
      }
   ],
   "NextToken": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Ledgers

Los libros de contabilidad que están asociados a la corriente Cuenta de AWS y a la región.

Tipo: matriz de objetos LedgerSummary

NextToken

Un token de paginación que indica si hay más resultados disponibles:

- Si `NextToken` está vacío, se ha procesado la última página de resultados y no hay más resultados que recuperar.
- Si `NextToken` no está vacío, hay más resultados disponibles. Para recuperar la siguiente página de resultados, use el valor de `NextToken` en una llamada posterior a `ListLedgers`.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

ListTagsForResource

Servicio: Amazon QLDB

Devuelve todas las etiquetas de un recurso de Amazon QLDB específico.

Sintaxis de la solicitud

```
GET /tags/resourceArn HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

resourceArn

El nombre de recurso de Amazon (ARN) del que se van a enumerar las etiquetas. Por ejemplo:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Tags

Devuelve las etiquetas actualmente asociadas al recurso de Amazon QLDB especificado.

Tipo: mapa de cadena a cadena

Entradas de mapa: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Limitaciones de longitud de la clave: longitud mínima de 1. Longitud máxima de 128.

Limitaciones de longitud de los valores: longitud mínima de 0. La longitud máxima es de 256 caracteres.

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2

- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

StreamJournalToKinesis

Servicio: Amazon QLDB

Crea una transmisión de diarios para un determinado libro mayor de Amazon QLDB. El flujo captura cada revisión del documento que está comprometida con el libro mayor y entrega los datos a un recurso de Amazon Kinesis Data Streams especificado.

Sintaxis de la solicitud

```
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"
   }
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

ExclusiveEndTime

La fecha y hora exclusivas que especifican cuándo termina la transmisión. Si no define este parámetro, la transmisión se ejecutará indefinidamente hasta que lo cancele.

La `ExclusiveEndTime` debe estar en formato de fecha y hora `ISO 8601` y en hora universal coordinada (UTC). Por ejemplo: `2019-06-13T21:36:34Z`.

Tipo: marca temporal

Obligatorio: no

InclusiveStartTime

Fecha y hora de inicio inclusivas a partir de la cual se iniciará la transmisión de datos del diario. Este parámetro debe estar en formato de fecha y hora `ISO 8601` y en hora universal coordinada (UTC). Por ejemplo: `2019-06-13T21:36:34Z`.

El `InclusiveStartTime` no puede estar en el futuro y debe ir antes de `ExclusiveEndTime`.

Si proporciona un `InclusiveStartTime` que sea anterior a la `CreationDateTime` de contabilidad, QLDB lo asigna por defecto a la `CreationDateTime` de contabilidad.

Tipo: marca temporal

Obligatorio: sí

KinesisConfiguration

Los ajustes de la configuración del destino de Kinesis Data Streams de su solicitud de transmisión.

Tipo: objeto KinesisConfiguration

Obligatorio: sí

RoleArn

El nombre de recurso de Amazon (ARN) del rol de IAM que concede a QLDB permisos para que una transmisión de diario escriba registros de datos en un recurso de Kinesis Data Streams.

Para transferir un rol a QLDB al solicitar una secuencia, debe tener permisos para realizar la acción `iam:PassRole` en el recurso de rol de IAM. Esto es necesario para todas las solicitudes de secuencias.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

StreamName

Nombre que desea asignar al flujo de diario QLDB. Los nombres definidos por el usuario pueden ayudar a identificar e indicar el propósito de un flujo.

El nombre del flujo debe ser único entre otros flujos activos de un libro mayor determinado. Los nombres de la transmisión tienen las mismas restricciones de denominación que los nombres de contabilidad, tal como se definen en Cuotas en Amazon QLDB en la Guía para desarrolladores de Amazon QLDB.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Tags

Los pares clave-valor que quiera agregar como etiquetas a la transmisión que desea crear. Las claves de etiqueta distinguen entre mayúsculas y minúsculas. Los valores de etiquetas distinguen entre mayúsculas y minúsculas y pueden ser nulos.

Tipo: mapa de cadena a cadena

Entradas de mapa: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Limitaciones de longitud de la clave: longitud mínima de 1. Longitud máxima de 128.

Limitaciones de longitud de los valores: longitud mínima de 0. La longitud máxima es de 256 caracteres.

Obligatorio: no

Sintaxis de la respuesta

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "StreamId": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

StreamId

El UUID (representado en texto codificado en Base62) que QLDB asigna a cada transmisión de diario QLDB.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

ResourcePreconditionNotMetException

La operación ha fallado porque no se ha cumplido una condición previa.

Código de estado HTTP: 412

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

TagResource

Servicio: Amazon QLDB

Añade una o varias etiquetas a un recurso especificado de Amazon QLDB.

Un recurso puede tener hasta 50 etiquetas. Si intenta crear más de 50 etiquetas para un recurso, la solicitud fallará y devolverá un error.

Sintaxis de la solicitud

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

resourceArn

El Nombre de recurso de Amazon (ARN) al que desea añadir las etiquetas. Por ejemplo:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

Tags

Los pares clave-valor que se agregarán como etiquetas al recurso QLDB especificado. Las claves de etiqueta distinguen entre mayúsculas y minúsculas. Si especifica una clave que ya existe para el recurso, la solicitud fallará y devolverá un error. Los valores de etiquetas distinguen entre mayúsculas y minúsculas y pueden ser nulos.

Tipo: mapa de cadena a cadena

Entradas de mapa: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Limitaciones de longitud de la clave: longitud mínima de 1. Longitud máxima de 128.

Limitaciones de longitud de los valores: longitud mínima de 0. La longitud máxima es de 256 caracteres.

Obligatorio: sí

Sintaxis de la respuesta

```
HTTP/1.1 200
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UntagResource

Servicio: Amazon QLDB

Elimina una o varias etiquetas del recurso de Amazon QLSB especificado. Puede especificar hasta 50 claves de etiqueta para eliminarlas.

Sintaxis de la solicitud

```
DELETE /tags/resourceArn?tagKeys=TagKeys HTTP/1.1
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

resourceArn

El nombre de recurso de Amazon (ARN) del que se van a eliminar etiquetas. Por ejemplo:

`arn:aws:qldb:us-east-1:123456789012:ledger/exampleLedger`

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

TagKeys

La lista de claves de etiquetas que se quieren eliminar.

Miembros de la matriz: número mínimo de 0 artículos. La cantidad máxima es de 200 artículos.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 128 caracteres.

Obligatorio: sí

Cuerpo de la solicitud

La solicitud no tiene un cuerpo de la solicitud.

Sintaxis de la respuesta

```
HTTP/1.1 200
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UpdateLedger

Servicio: Amazon QLDB

Actualiza las propiedades de un libro mayor.

Sintaxis de la solicitud

```
PATCH /ledgers/name HTTP/1.1
Content-type: application/json

{
   "DeletionProtection": boolean,
   "KmsKey": "string"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

DeletionProtection

Especifica si el libro mayor está protegido contra la eliminación por parte de cualquier usuario. Si no se define al crear el libro mayor, esta característica estará habilitada (`true`) de forma predeterminada.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede llevar a cabo la desactivación mediante una llamada a la operación `UpdateLedger` para establecer este parámetro en `false`.

Tipo: Booleano

Obligatorio: no

KmsKey

La clave in AWS Key Management Service (AWS KMS) que se utilizará para cifrar los datos en reposo en el libro mayor. Para obtener más información, consulte Encryption at rest (Cifrado en reposo) en la Guía para desarrolladores de Amazon QLDB.

Utilice una de las siguientes opciones para especificar este parámetro:

- `AWS_OWNED_KMS_KEY`: utilice una AWS KMS clave que sea de su propiedad y que esté AWS gestionada por usted.
- Sin definir: no realiza cambios en la clave KMS del libro mayor.
- Una clave KMS válida y simétrica que administra el cliente: utilice la clave KMS de cifrado simétrico especificada en la cuenta que ha creado y que posee y administra.

  Amazon QLDB no es compatible con claves asimétricas. Para obtener más información, consulte Uso de claves simétricas y asimétricas en la Guía para AWS Key Management Service desarrolladores.

Para especificar una clave KMS que administre el cliente, utilice el ID de clave, el nombre de recurso de Amazon (ARN), el nombre de alias o bien el ARN de alias. Cuando utilice un nombre de alias, utilice el prefijo `"alias/"`. Para especificar una clave en otra Cuenta de AWS, debe usar la clave ARN o el alias ARN.

Por ejemplo:

- ID de clave: `1234abcd-12ab-34cd-56ef-1234567890ab`
- ARN de clave: `arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nombre de alias: `alias/ExampleAlias`
- ARN de alias: `arn:aws:kms:us-east-2:111122223333:alias/ExampleAlias`

Para obtener más información, consulte los identificadores clave (KeyId) en la AWS Key Management Service Guía para desarrolladores.

Tipo: cadena

Limitaciones de longitud: longitud máxima de 1600 caracteres.

Obligatorio: no

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json

{
   "Arn": "string",
   "CreationDateTime": number,
   "DeletionProtection": boolean,
   "EncryptionDescription": {
      "EncryptionStatus": "string",
      "InaccessibleKmsKeyDateTime": number,
      "KmsKeyArn": "string"
   },
   "Name": "string",
   "State": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Arn

El nombre de recurso de Amazon (ARN) para el libro mayor.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

CreationDateTime

La fecha y la hora, en formato de tiempo epoch, en que se creó el libro mayor. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Tipo: marca temporal

DeletionProtection

Especifica si el libro mayor está protegido contra la eliminación por parte de cualquier usuario. Si no se define al crear el libro mayor, esta característica estará habilitada (`true`) de forma predeterminada.

Si la protección contra eliminación está habilitada, primero debe desactivarla para poder eliminar el libro mayor. Puede llevar a cabo la desactivación mediante una llamada a la operación `UpdateLedger` para establecer este parámetro en `false`.

Tipo: Booleano

EncryptionDescription

Información sobre el cifrado de los datos en reposo del libro mayor. Esto incluye el estado actual, la AWS KMS clave y cuándo dejó de estar accesible la clave (en caso de error).

Tipo: objeto LedgerEncryptionDescription

Name

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

State

El estado actual del libro mayor.

Tipo: cadena

Valores válidos: `CREATING | ACTIVE | DELETING | DELETED`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

UpdateLedgerPermissionsMode

Servicio: Amazon QLDB

Actualiza el modo de permisos del libro mayor.

> **Important**
>
> Antes de cambiar al modo de permisos `STANDARD`, deberá crear todas las políticas de IAM y etiquetas de tabla necesarias para evitar interrupciones a los usuarios. Para obtener más información, consulte Migrating to the standard permissions mode en la Guía para desarrolladores de Amazon QLDB.

Sintaxis de la solicitud

```
PATCH /ledgers/name/permissions-mode HTTP/1.1
Content-type: application/json

{
   "PermissionsMode": "string"
}
```

Parámetros de solicitud del URI

La solicitud utiliza los siguientes parámetros URI.

name

El nombre de contabilidad.

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Cuerpo de la solicitud

La solicitud acepta los siguientes datos en formato JSON.

PermissionsMode

El modo de permisos que se va a asignar al libro mayor. Este parámetro puede tener uno de los siguientes valores:

- `ALLOW_ALL`: un modo de permisos heredado que habilita el control de acceso con granularidad de la API para los libros mayores.

  Este modo permite a los usuarios que tengan el permiso de API `SendCommand` para este libro mayor poner en marcha todos los comandos de PartiQL (por lo tanto, `ALLOW_ALL`) en cualquier tabla del libro mayor especificado. Este modo no tendrá en cuenta las políticas de permisos de IAM de la tabla o comando que cree para el libro mayor.

- `STANDARD`: (recomendado) un modo de permisos que habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL.

  De forma predeterminada, este modo deniega todas las solicitudes de los usuarios para poner en marcha cualquier comando de PartiQL en tablas de este libro mayor. Para permitir que se pongan en marcha comandos de PartiQL, debe crear políticas de permisos de IAM para recursos de tablas y acciones de PartiQL específicos, además del permiso de API `SendCommand` para el libro mayor. Para obtener más información, consulte Getting started with the standard permissions mode (Introducción al modo de permisos estándar) en la Guía para desarrolladores de Amazon QLDB.

> **Note**
>
> Recomendamos encarecidamente el modo `STANDARD` de permisos para maximizar la seguridad de los datos del libro mayor.

Tipo: cadena

Valores válidos: `ALLOW_ALL | STANDARD`

Obligatorio: sí

Sintaxis de la respuesta

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Arn": "string",
   "Name": "string",
   "PermissionsMode": "string"
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

Arn

El nombre de recurso de Amazon (ARN) para el libro mayor.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Name

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

PermissionsMode

El modo actual de permisos del libro mayor.

Tipo: cadena

Valores válidos: `ALLOW_ALL | STANDARD`

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

InvalidParameterException

Uno o más parámetros de la solicitud no son válidos.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado no existe.

Código de estado HTTP: 404

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3
- AWS SDK para Python
- AWS SDK para Ruby V3

Sesión de Amazon QLDB

La sesión de Amazon QLDB admite las siguientes acciones:

- SendCommand

SendCommand

Servicio: Amazon QLDB Session

Envía un comando a un libro mayor de Amazon QLDB.

> **Note**
>
> En lugar de interactuar directamente con esta API, recomendamos usar el controlador de QLDB o el intérprete de comandos de QLDB para ejecutar transacciones de datos en un libro mayor.
>
> - Si trabaja con un AWS SDK, utilice el controlador QLDB. El controlador proporciona una capa de abstracción de alto nivel sobre la API de datos de sesión de QLDB, y gestiona la operación de `SendCommand` por usted. Para obtener más información y una lista de los lenguajes de programación compatibles, consulte Introducción al controlador en la Guía del desarrollador de Amazon QLDB.
> - Si está trabajando con AWS Command Line Interface (AWS CLI), utilice el shell QLDB. El intérprete de comandos es una interfaz de la línea de comandos que usa el controlador de QLDB para interactuar con un libro mayor. Para obtener más información, consulte Acceder a Amazon QLDB mediante el intérprete de comandos de QLDB.

Sintaxis de la solicitud

```
{
   "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": {
   }
}
```

Parámetros de la solicitud

Para obtener información sobre los parámetros comunes a todas las acciones, consulte Parámetros comunes.

La solicitud acepta los siguientes datos en formato JSON.

AbortTransaction

Comando para anular la transacción actual.

Tipo: objeto AbortTransactionRequest

Obligatorio: no

CommitTransaction

Comando para confirmar la transacción especificada.

Tipo: objeto CommitTransactionRequest

Obligatorio: no

EndSession

Comando para finalizar la sesión actual.

Tipo: objeto EndSessionRequest

Obligatorio: no

ExecuteStatement

Comando para ejecutar una instrucción en la transacción especificada.

Tipo: objeto ExecuteStatementRequest

Obligatorio: no

FetchPage

Comando para recuperar una página.

Tipo: objeto FetchPageRequest

Obligatorio: no

SessionToken

Especifica el token de sesión del comando actual. Un token de sesión es constante a lo largo de toda la sesión.

Para obtener un token de sesión, ejecute el comando `StartSession`. Este `SessionToken` será necesario para todos los comandos posteriores que se emitan durante la sesión actual.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Obligatorio: no

StartSession

Comando para iniciar una nueva sesión. Se obtiene un token de sesión como parte de la respuesta.

Tipo: objeto StartSessionRequest

Obligatorio: no

StartTransaction

Comando para iniciar una nueva transacción.

Tipo: objeto StartTransactionRequest

Obligatorio: no

Sintaxis de la respuesta

```
{
   "AbortTransaction": {
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "CommitTransaction": {
      "CommitDigest": blob,
      "ConsumedIOs": {
         "ReadIOs": number,
         "WriteIOs": number
      },
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      },
      "TransactionId": "string"
   },
   "EndSession": {
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "ExecuteStatement": {
      "ConsumedIOs": {
         "ReadIOs": number,
         "WriteIOs": number
      },
      "FirstPage": {
         "NextPageToken": "string",
         "Values": [
            {
               "IonBinary": blob,
               "IonText": "string"
            }
         ]
      },
      "TimingInformation": {
         "ProcessingTimeMilliseconds": number
      }
   },
   "FetchPage": {
      "ConsumedIOs": {
```

```
            "ReadIOs": number,
            "WriteIOs": number
         },
         "Page": {
            "NextPageToken": "string",
            "Values": [
               {
                  "IonBinary": blob,
                  "IonText": "string"
               }
            ]
         },
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         }
      },
      "StartSession": {
         "SessionToken": "string",
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         }
      },
      "StartTransaction": {
         "TimingInformation": {
            "ProcessingTimeMilliseconds": number
         },
         "TransactionId": "string"
      }
}
```

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

AbortTransaction

Contiene los detalles de la transacción anulada.

Tipo: objeto AbortTransactionResult

CommitTransaction

Contiene los detalles de la transacción confirmada.

Tipo: objeto CommitTransactionResult

EndSession

Contiene los detalles de la sesión finalizada.

Tipo: objeto EndSessionResult

ExecuteStatement

Contiene los detalles de la instrucción ejecutada.

Tipo: objeto ExecuteStatementResult

FetchPage

Contiene los detalles de la página recuperada.

Tipo: objeto FetchPageResult

StartSession

Contiene los detalles de la sesión iniciada, que incluye un token de sesión. Este `SessionToken` será necesario para todos los comandos posteriores que se emitan durante la sesión actual.

Tipo: objeto StartSessionResult

StartTransaction

Contiene los detalles de la transacción iniciada.

Tipo: objeto StartTransactionResult

Errores

Para obtener información acerca de los errores comunes a todas las acciones, consulte Errores comunes.

BadRequestException

Se devuelve si la solicitud tiene un formato incorrecto o contiene un error, como un valor de parámetro no válido o falta de parámetro obligatorio.

Código de estado HTTP: 400

CapacityExceededException

Se devuelve cuando la solicitud supera la capacidad de procesamiento del libro mayor.

Código de estado HTTP: 400

InvalidSessionException

Se devuelve si la sesión ya no existe porque se agotó el tiempo de espera o caducó.

Código de estado HTTP: 400

LimitExceededException

Se devuelve si se supera un límite de recursos, como el número de sesiones activas.

Código de estado HTTP: 400

OccConflictException

Se devuelve cuando no se puede escribir una transacción en el diario debido a un fallo en la fase de verificación del control de concurrencia optimista (OCC).

Código de estado HTTP: 400

RateExceededException

Devuelto cuando la tasa de solicitudes excede el rendimiento permitido.

Código de estado HTTP: 400

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- Interfaz de la línea de comandos de AWS
- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go v2
- AWS SDK para Java V2
- AWS SDK para JavaScript V3
- AWS SDK para PHP V3

- AWS SDK para Python
- AWS SDK para Ruby V3

Tipos de datos

Amazon QLDB admite los siguientes tipos de datos:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

La sesión de Amazon QLDB admite los siguientes tipos de datos:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult
- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult

- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

Amazon QLDB

Amazon QLDB admite los siguientes tipos de datos:

- JournalKinesisStreamDescription
- JournalS3ExportDescription
- KinesisConfiguration
- LedgerEncryptionDescription
- LedgerSummary
- S3EncryptionConfiguration
- S3ExportConfiguration
- ValueHolder

JournalKinesisStreamDescription

Servicio: Amazon QLDB

Información sobre la secuencia del diario de Amazon QLDB. El resultado incluye el nombre de recurso de Amazon (ARN), el nombre de secuencia, la hora de creación, el estado actual y los parámetros de la solicitud de creación de secuencia original.

Contenido

KinesisConfiguration

Los valores de configuración del destino de Amazon Kinesis Data Streams para un flujo del diario de QLDB.

Tipo: objeto KinesisConfiguration

Obligatorio: sí

LedgerName

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

RoleArn

El nombre de recurso de Amazon (ARN) del rol de IAM que concede a QLDB permisos para que una transmisión de diario escriba registros de datos en un recurso de Kinesis Data Streams.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

Status

El estado actual de la secuencia del diario QLDB.

Tipo: cadena

Valores válidos: `ACTIVE | COMPLETED | CANCELED | FAILED | IMPAIRED`

Obligatorio: sí

StreamId

El UUID (representado en texto codificado en Base62) de la secuencia del diario QLDB.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

StreamName

Nombre definido por el usuario de la secuencia del diario QLDB.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Arn

Nombre de recurso de Amazon (ARN) de la secuencia del diario QLDB.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: no

CreationTime

La fecha y la hora, en formato de tiempo epoch, en que se creó la secuencia del diario QLDB. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Tipo: marca temporal

Obligatorio: no

ErrorCause

El mensaje de error que describe el motivo por el que una secuencia tiene el estado de `IMPAIRED` o `FAILED`. Esto no se aplica a las secuencias que tienen otros valores de estado.

Tipo: cadena

Valores válidos: `KINESIS_STREAM_NOT_FOUND | IAM_PERMISSION_REVOKED`

Obligatorio: no

ExclusiveEndTime

La fecha y hora exclusivas que especifican cuándo termina la transmisión. Si no define este parámetro, la secuencia se ejecutará indefinidamente hasta que lo cancele.

Tipo: marca temporal

Obligatorio: no

InclusiveStartTime

Fecha y hora de inicio inclusivas a partir de la cual se iniciará la transmisión de datos del diario.

Tipo: marca temporal

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

JournalS3ExportDescription

Servicio: Amazon QLDB

Información sobre un trabajo de exportación de diario que incluye el nombre del libro mayor, la ID de exportación, la hora de creación, el estado actual y los parámetros de la solicitud de creación de exportación original.

Contenido

ExclusiveEndTime

La fecha y hora de finalización exclusivas del rango de contenidos del diario especificado en la solicitud de exportación original.

Tipo: marca temporal

Obligatorio: sí

ExportCreationTime

La fecha y la hora, en formato de tiempo Unix, en que se creó el trabajo de exportación. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Tipo: marca temporal

Obligatorio: sí

ExportId

El UUID (representado en texto codificado en Base62) del trabajo de exportación del diario.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

InclusiveStartTime

La fecha y hora de inicio inclusivas del rango de contenidos del diario que se especificó en la solicitud de exportación original.

Tipo: marca temporal

Obligatorio: sí

LedgerName

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

RoleArn

El nombre de recurso de Amazon (ARN) del rol de IAM que concede permisos a QLDB para que los trabajos de exportación de diarios hagan lo siguiente:

- Escriba objetos en su bucket de Amazon Simple Storage Service (Amazon S3).
- (Opcional) Utilice su clave gestionada por el cliente AWS Key Management Service (AWS KMS) para cifrar los datos exportados en el servidor.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

S3ExportConfiguration

Ubicación del bucket de Amazon Simple Storage Service (Amazon S3) en el que el trabajo de exportación del diario escribe el contenido del diario.

Tipo: objeto S3ExportConfiguration

Obligatorio: sí

Status

El estado actual del trabajo de exportación del diario.

Tipo: cadena

Valores válidos: `IN_PROGRESS | COMPLETED | CANCELLED`

Obligatorio: sí

OutputFormat

El formato de salida de los datos del diario exportado.

Tipo: cadena

Valores válidos: `ION_BINARY | ION_TEXT | JSON`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

KinesisConfiguration

Servicio: Amazon QLDB

Los valores de configuración del destino de Amazon Kinesis Data Streams para un flujo del diario de Amazon QLDB.

Contenido

StreamArn

Anote el nombre de recurso de Amazon (ARN) de Kinesis Data Streams.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

AggregationEnabled

Permite a QLDB publicar varios registros de datos en un único registro de Kinesis Data Streams, lo que aumenta el número de registros enviados por llamada a la API.

Valor predeterminado: `True`

> **Important**
>
> La agregación de registros tiene importantes implicaciones para el procesamiento de registros y requiere la desagrupación en su consumidor de flujos. Para obtener más información, consulte Conceptos clave de KPL y Desagrupación del consumidor en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Tipo: Booleano

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LedgerEncryptionDescription

Servicio: Amazon QLDB

Información sobre el cifrado de los datos en reposo del libro mayor de Amazon QLDB. Esto incluye el estado actual, la clave en AWS Key Management Service (AWS KMS) y cuándo se volvió inaccesible la clave (en caso de error).

Para obtener más información, consulte Encryption at rest (Cifrado en reposo) en la Guía para desarrolladores de Amazon QLDB.

Contenido

EncryptionStatus

Estado actual del cifrado en reposo del libro mayor. Puede ser uno de los valores siguientes:

- `ENABLED`: el cifrado está totalmente activado con la clave especificada.
- `UPDATING`: el libro mayor está procesando activamente el cambio de clave especificado.

  Los cambios de clave en QLDB son asíncronos. Mientras se procesa el cambio de clave, el registro es totalmente accesible sin ningún impacto en el rendimiento. El tiempo que tarda en actualizarse una clave varía según el tamaño del libro mayor.
- `KMS_KEY_INACCESSIBLE`: no se puede acceder a la clave KMS administrada por el cliente, y el libro mayor está dañado. La clave se ha deshabilitado o eliminado, o bien se han revocado las concesiones de la clave. Cuando un libro mayor está dañado, no es accesible y no acepta solicitudes de lectura ni escritura.

  Un libro mayor dañado regresa automáticamente a su estado activo tras restablecer las concesiones de la clave o reactivar la clave que estaba desactivada. Sin embargo, la eliminación de una clave KMS administrada por el cliente es irreversible. Una vez que se elimina una clave, ya no puede acceder a los libros mayores que están protegidos con ella y los datos se vuelven irrecuperables permanentemente.

Tipo: cadena

Valores válidos: `ENABLED | UPDATING | KMS_KEY_INACCESSIBLE`

Obligatorio: sí

KmsKeyArn

El nombre de recurso de Amazon (ARN) de la clave KMS administrada por el cliente que emplea el libro mayor para el cifrado en reposo. Si este parámetro no está definido, el libro mayor utiliza una clave KMS propia AWS para el cifrado. Se mostrará `AWS_OWNED_KMS_KEY` al actualizar la configuración de cifrado del libro mayor a la clave KMS propia AWS .

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: sí

InaccessibleKmsKeyDateTime

La fecha y la hora, en formato de época y hora, en las que la AWS KMS clave quedó inaccesible por primera vez, en caso de error. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Este parámetro no está definido si se puede acceder a la AWS KMS clave.

Tipo: marca temporal

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

LedgerSummary

Servicio: Amazon QLDB

Información sobre un libro mayor, incluidos su nombre, estado y fecha de creación.

Contenido

CreationDateTime

La fecha y la hora, en formato de tiempo epoch, en que se creó el libro mayor. (El formato de tiempo Epoch es el número de segundos transcurridos desde las 12:00:00 a.m. del 1 de enero de 1970 en UTC.)

Tipo: marca temporal

Obligatorio: no

Name

El nombre de contabilidad.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: no

State

El estado actual del libro mayor.

Tipo: cadena

Valores válidos: `CREATING | ACTIVE | DELETING | DELETED`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

S3EncryptionConfiguration

Servicio: Amazon QLDB

Los ajustes de cifrado que emplea un trabajo de exportación de diario para escribir datos en un bucket de Amazon Simple Storage Service (Amazon S3).

Contenido

ObjectEncryptionType

El tipo de cifrado de objetos de Amazon S3.

Para obtener más información sobre opciones de cifrado del servidor en Amazon S3, consulte Protección de datos mediante cifrado del servidor en la Guía para desarrolladores de Amazon S3.

Tipo: cadena

Valores válidos: `SSE_KMS | SSE_S3 | NO_ENCRYPTION`

Obligatorio: sí

KmsKeyArn

El nombre de recurso de Amazon (ARN) de una clave de cifrado simétrica en AWS Key Management Service ().AWS KMS Amazon S3 no es compatible con claves de KMS asimétricas.

Debe proporcionar un `KmsKeyArn` si especifica `SSE_KMS` como `ObjectEncryptionType`.

Si especifica `SSE_S3` como el `ObjectEncryptionType`, entonces `KmsKeyArn` no es obligatorio.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 20. La longitud máxima es de 1600 caracteres.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulte lo siguiente:

- AWS SDK para C++

- AWS SDK para Java V2
- AWS SDK para Ruby V3

S3ExportConfiguration

Servicio: Amazon QLDB

Ubicación del bucket de Amazon Simple Storage Service (Amazon S3) en el que el trabajo de exportación del diario escribe el contenido del diario.

Contenido

Bucket

El nombre del bucket de Amazon S3 en el que el trabajo de exportación del diario escribe el contenido del diario.

El nombre del bucket debe cumplir con las convenciones de nomenclatura de buckets de Amazon S3. Para obtener más información, consulte Restricciones y limitaciones de los buckets en la Guía para desarrolladores de Amazon S3.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 3. La longitud máxima es de 255 caracteres.

Patrón: `^[A-Za-z-0-9-_.]+$`

Obligatorio: sí

EncryptionConfiguration

Los ajustes de cifrado que emplea un trabajo de exportación de diario para escribir datos en un bucket de Amazon S3.

Tipo: objeto S3EncryptionConfiguration

Obligatorio: sí

Prefix

Prefijo del bucket de Amazon S3 en el que el trabajo de exportación de diario escribe el contenido del diario.

El prefijo debe cumplir con las reglas y restricciones de nomenclatura de claves de Amazon S3. Para obtener más información, consulte Claves y metadatos de objetos en la Guía para desarrolladores de Amazon S3.

Los siguientes son ejemplos de valores de `Prefix` válidos:

- `JournalExports-ForMyLedger/Testing/`
- `JournalExports`
- `My:Tests/`

Tipo: cadena

Limitaciones de longitud: longitud mínima de 0. La longitud máxima es de 128 caracteres.

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ValueHolder

Servicio: Amazon QLDB

Estructura que puede contener un valor en múltiples formatos de codificación.

Contenido

IonText

Un valor de texto plano de Amazon Ion contenido en una estructura `ValueHolder`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1048576.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Sesión de Amazon QLDB

La sesión de Amazon QLDB admite los siguientes tipos de datos:

- AbortTransactionRequest
- AbortTransactionResult
- CommitTransactionRequest
- CommitTransactionResult
- EndSessionRequest
- EndSessionResult
- ExecuteStatementRequest
- ExecuteStatementResult

- FetchPageRequest
- FetchPageResult
- IOUsage
- Page
- StartSessionRequest
- StartSessionResult
- StartTransactionRequest
- StartTransactionResult
- TimingInformation
- ValueHolder

AbortTransactionRequest

Servicio: Amazon QLDB Session

Contiene los detalles de la transacción que se va a anular.

Contenido

Los miembros de esta estructura de excepción dependen del contexto.

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

AbortTransactionResult

Servicio: Amazon QLDB Session

Contiene los detalles de la transacción anulada.

Contenido

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

CommitTransactionRequest

Servicio: Amazon QLDB Session

Contiene los detalles de la transacción que se va a confirmar.

Contenido

CommitDigest

Especifica el resumen de confirmación de la transacción que se va a confirmar. Para cada transacción activa, se debe enviar el resumen de confirmaciones. QLDB valida `CommitDigest` y rechaza la confirmación con un error si el resumen calculado en el cliente no coincide con el resumen calculado por QLDB.

El propósito del parámetro `CommitDigest` es garantizar que la QLDB confirme una transacción si y solo si el servidor ha procesado el conjunto exacto de instrucciones enviadas por el cliente, en el mismo orden en que las envió el cliente y sin duplicados.

Tipo: objeto de datos binarios codificados en Base64

Obligatorio: sí

TransactionId

Especifica el ID de la transacción que desea confirmar.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

CommitTransactionResult

Servicio: Amazon QLDB Session

Contiene los detalles de la transacción confirmada.

Contenido

CommitDigest

El resumen de confirmación de la transacción confirmada.

Tipo: objeto de datos binarios codificados en Base64

Obligatorio: no

ConsumedIOs

Contiene métricas sobre el número de solicitudes de E/S que se consumieron.

Tipo: objeto IOUsage

Obligatorio: no

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

TransactionId

Especifica el ID de la transacción confirmada.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

EndSessionRequest

Servicio: Amazon QLDB Session

Especifica una solicitud para finalizar la sesión.

Contenido

Los miembros de esta estructura de excepción dependen del contexto.

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

EndSessionResult

Servicio: Amazon QLDB Session

Contiene los detalles de la sesión finalizada.

Contenido

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ExecuteStatementRequest

Servicio: Amazon QLDB Session

Especifica una solicitud para ejecutar una instrucción.

Contenido

Statement

Especifica la instrucción de la solicitud.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 100000.

Obligatorio: sí

TransactionId

Especifica el identificador de la transacción de la solicitud.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

Parameters

Especifica los parámetros de la instrucción parametrizada de la solicitud.

Tipo: matriz de objetos ValueHolder

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2

- AWS SDK para Ruby V3

ExecuteStatementResult

Servicio: Amazon QLDB Session

Contiene los detalles de la instrucción ejecutada.

Contenido

ConsumedIOs

Contiene métricas sobre el número de solicitudes de E/S que se consumieron.

Tipo: objeto IOUsage

Obligatorio: no

FirstPage

Contiene los detalles de la primera página recuperada.

Tipo: objeto Page

Obligatorio: no

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

FetchPageRequest

Servicio: Amazon QLDB Session

Especifica los detalles de la página que se va a buscar.

Contenido

NextPageToken

Especifica el token de página siguiente de la página que se va a buscar.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Obligatorio: sí

TransactionId

Especifica el ID de transacción de la página que se va a buscar.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

FetchPageResult

Servicio: Amazon QLDB Session

Contiene la página que se recuperó.

Contenido

ConsumedIOs

Contiene métricas sobre el número de solicitudes de E/S que se consumieron.

Tipo: objeto IOUsage

Obligatorio: no

Page

Contiene los detalles de la página recuperada.

Tipo: objeto Page

Obligatorio: no

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

IOUsage

Servicio: Amazon QLDB Session

Contiene las métricas de uso de E/S de un comando que se ha invocado.

Contenido

ReadIOs

Es el número de solicitudes de E/S de lectura realizadas por el comando.

Tipo: largo

Obligatorio: no

WriteIOs

Es el número de solicitudes de E/S de escritura realizadas por el comando.

Tipo: largo

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Page

Servicio: Amazon QLDB Session

Contiene los detalles de la página recuperada.

Contenido

NextPageToken

El token de la página siguiente.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Obligatorio: no

Values

Estructura que contiene valores en múltiples formatos de codificación.

Tipo: matriz de objetos ValueHolder

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartSessionRequest

Servicio: Amazon QLDB Session

Especifica una solicitud para iniciar una nueva sesión.

Contenido

LedgerName

El nombre del libro mayor con el que iniciar una nueva sesión.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. La longitud máxima es de 32 caracteres.

Patrón: `(?!^.*--)(?!^[0-9]+$)(?!^-)(?!.*-$)^[A-Za-z0-9-]+$`

Obligatorio: sí

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartSessionResult

Servicio: Amazon QLDB Session

Contiene los detalles de la sesión iniciada.

Contenido

SessionToken

Token de sesión de la sesión iniciada. Este `SessionToken` será necesario para todos los comandos posteriores que se emitan durante la sesión actual.

Tipo: cadena

Limitaciones de longitud: longitud mínima de 4. La longitud máxima es de 1024 caracteres.

Patrón: `^[A-Za-z-0-9+/=]+$`

Obligatorio: no

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartTransactionRequest

Servicio: Amazon QLDB Session

Especifica una solicitud para iniciar una transacción.

Contenido

Los miembros de esta estructura de excepción dependen del contexto.

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

StartTransactionResult

Servicio: Amazon QLDB Session

Contiene los detalles de la transacción iniciada.

Contenido

TimingInformation

Contiene información sobre el rendimiento del comando del servidor.

Tipo: objeto TimingInformation

Obligatorio: no

TransactionId

ID de transacción de la transacción iniciada.

Tipo: cadena

Limitaciones de longitud: longitud fija de 22.

Patrón: `^[A-Za-z-0-9]+$`

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

TimingInformation

Servicio: Amazon QLDB Session

Contiene información sobre el rendimiento de un comando del servidor. Amazon QLDB registra la información de temporización entre el momento en que recibe la solicitud y el momento en que envía la respuesta correspondiente.

Contenido

ProcessingTimeMilliseconds

La cantidad de tiempo que QLDB ha dedicado a procesar el comando, indicada en milisegundos.

Tipo: largo

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

ValueHolder

Servicio: Amazon QLDB Session

Estructura que puede contener un valor en múltiples formatos de codificación.

Contenido

IonBinary

Un valor binario de Amazon Ion contenido en una estructura `ValueHolder`.

Tipo: objeto de datos binarios codificados en Base64

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 131072.

Obligatorio: no

IonText

Un valor de texto plano de Amazon Ion contenido en una estructura `ValueHolder`.

Tipo: string

Limitaciones de longitud: longitud mínima de 1. Longitud máxima de 1048576.

Obligatorio: no

Véase también

Para obtener más información sobre el uso de esta API en uno de los AWS SDK específicos del idioma, consulta lo siguiente:

- AWS SDK para C++
- AWS SDK para Java V2
- AWS SDK para Ruby V3

Errores comunes

En esta sección, se enumeran los errores comunes a las acciones de la API de todos los servicios de AWS. En el caso de los errores específicos de una acción de la API de este servicio, consulte el tema de dicha acción de la API.

AccessDeniedException

No tiene acceso suficiente para realizar esta acción.

Código de estado HTTP: 400

IncompleteSignature

La firma de solicitud no se ajusta a los estándares de AWS.

Código de estado HTTP: 400

InternalFailure

El procesamiento de la solicitud ha devuelto un error debido a un error o una excepción desconocidos.

Código de estado HTTP: 500

InvalidAction

La acción u operación solicitada no es válida. Compruebe que la acción se ha escrito correctamente.

Código de estado HTTP: 400

InvalidClientTokenId

El certificado X.509 o el ID de clave de acceso de AWS proporcionado no existen en nuestros registros.

Código de estado HTTP: 403

NotAuthorized

No tiene permiso para realizar esta acción.

Código de estado HTTP: 400

OptInRequired

El ID de clave de acceso de AWS necesita una suscripción al servicio.

Código de estado HTTP: 403

RequestExpired

La solicitud llegó al servicio más de 15 minutos después de la marca de fecha en la solicitud o más de 15 minutos después de la fecha de vencimiento de la solicitud (por ejemplo, para las URL

prefirmadas) o la marca de fecha de la solicitud corresponde a una hora futura en más de 15 minutos.

Código de estado HTTP: 400

ServiceUnavailable

La solicitud no se ha ejecutado correctamente debido a un error temporal del servidor.

Código de estado HTTP: 503

ThrottlingException

La solicitud se denegó debido a una limitación controlada.

Código de estado HTTP: 400

ValidationError

La entrada no satisface las limitaciones que especifica un servicio de AWS.

Código de estado HTTP: 400

Parámetros comunes

La siguiente lista contiene los parámetros que utilizan todas las acciones para firmar solicitudes de Signature Version 4 con una cadena de consulta. Los parámetros específicos de acción se enumeran en el tema correspondiente a la acción. Para obtener más información sobre Signature Version 4, consulte Firma de solicitudes API de AWS en la Guía del usuario de IAM.

Action

Las acciones que se van a realizar.

Tipo: cadena

Obligatorio: sí

Version

La versión de la API para la que está escrita la solicitud, expresada en el formato AAAA-MM-DD.

Tipo: String

Obligatorio: sí

X-Amz-Algorithm

El algoritmo de hash que utilizó para crear la solicitud de firma.

Condición: especifique este parámetro cuando incluya información de autenticación en una cadena de consulta en lugar de en el encabezado de autorización HTTP.

Tipo: String

Valores válidos: `AWS4-HMAC-SHA256`

Obligatorio: condicional

X-Amz-Credential

El valor del ámbito de la credencial, que es una cadena que incluye la clave de acceso, la fecha, la región a la que se dirige, el servicio que solicita y una cadena de terminación (“aws4_request”). El valor se expresa en el siguiente formato: access_key/AAAAMMDD/region/service/aws4_request.

Para obtener más información, consulte Crear una solicitud API de AWS firmada en la Guía del usuario de IAM.

Condición: especifique este parámetro cuando incluya información de autenticación en una cadena de consulta en lugar de en el encabezado de autorización HTTP.

Tipo: cadena

Obligatorio: condicional

X-Amz-Date

La fecha utilizada para crear la firma. El formato debe ser ISO 8601 formato básico (AAAAMMDD‘T’HHMMSS‘Z’). Por ejemplo, la siguiente fecha y hora es un valor válido de X-Amz-Date para `20120325T120000Z`.

Condición: X-Amz-Date es opcional en todas las solicitudes; se puede utilizar para anular la fecha empleada a fin de firmar las solicitudes. Si el encabezado Date se especifica en el formato básico ISO 8601, no se requiere X-Amz-Date. Cuando se usa X-Amz-Date, siempre anula el valor del encabezado Date. Para obtener más información, consulte Elementos de una firma de solicitud API de AWS en la Guía del usuario de IAM.

Tipo: cadena

Obligatorio: condicional

X-Amz-Security-Token

El token de seguridad temporal que se obtuvo mediante una llamada a AWS Security Token Service (AWS STS). Para obtener una lista de servicios compatibles con las credenciales de seguridad temporales de AWS STS, consulte Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Condición: si utiliza credenciales de seguridad temporales de AWS STS, debe incluir el token de seguridad.

Tipo: cadena

Obligatorio: condicional

X-Amz-Signature

Especifica la firma codificada hexadecimal que se calculó a partir de la cadena que se va a firmar y la clave de firma derivada.

Condición: especifique este parámetro cuando incluya información de autenticación en una cadena de consulta en lugar de en el encabezado de autorización HTTP.

Tipo: cadena

Obligatorio: condicional

X-Amz-SignedHeaders

Especifica todos los encabezados HTTP que se incluyeron como parte de la solicitud canónica. Para obtener más información acerca de especificar encabezados firmados, consulte Crear una solicitud API de AWS firmada en la Guía del usuario de IAM.

Condición: especifique este parámetro cuando incluya información de autenticación en una cadena de consulta en lugar de en el encabezado de autorización HTTP.

Tipo: cadena

Obligatorio: condicional

Cuotas y límites de Amazon QLDB

En esta sección se describen las cuotas actuales, también denominadas límites, de Amazon QLDB.

Temas

Cuotas predeterminadas

QLDB tiene las siguientes cuotas predeterminadas, que también figuran en los puntos de conexión y cuotas de Amazon QLDB en Referencia general de AWS. Estas cuotas se aplican por Cuenta de AWS y por región. Para solicitar un aumento de una cuota para su cuenta en una región, use la consola de Service Quotas.

Inicie sesión en la AWS Management Console y abra la consola de Service Quotas en https://console.aws.amazon.com/servicequotas/.

Recurso	Cuota predeterminada
El número máximo de libros mayores activos que puede crear en esta cuenta en la región actual	5
El número máximo de exportaciones de diarios activos a Amazon S3 por libro mayor	2
El número máximo de secuencias de diario activas a Kinesis Data Streams por libro mayor	5

Cuotas fijas

Además de las cuotas predeterminadas, QLDB tiene las siguientes cuotas fijas por libro mayor. Estas cuotas no se pueden aumentar mediante Service Quotas:

Recurso	Cuota fija
Número de sesiones activas simultáneas	1500
Número de tablas activas	20
Número total de tablas (activas e inactivas) **Note** En QLDB, las tablas eliminadas se consideran inactivas y se tienen en cuenta para esta cuota total.	40
Número de índices por tabla	5
Número de documentos de una transacción	40
Número de revisiones que se van a editar en una transacción	1
Tamaño del documento (codificado en formato `IonBinary`)	128 KB
Tamaño del parámetro de la instrucción (formato `IonBinary`)	128 KB
Tamaño del parámetro de la instrucción (formato `IonText`)	1 MB
Longitud de cadena de instrucción	100 000 caracteres
Tamaño de transacción	4 MB

Recurso	Cuota fija
Tiempo de espera de transacción	30 segundos
Periodo de caducidad de los trabajos de exportación de diarios completados	7 días
Periodo de caducidad de las secuencias de diarios terminales	7 días

Cuota del libro mayor

Para solicitar un aumento de una cuota de libro mayor para su cuenta en una región, use la consola de Service Quotas.

Abra la consola de Service Quotas en https://console.aws.amazon.com/servicequotas/.

Algunos casos de uso de la QLDB requieren un número cada vez mayor de libros mayores por Cuenta de AWS y por región en función del crecimiento empresarial. Por ejemplo, es posible que necesite crear libros mayores específicos para aislar a los clientes o los datos. En este caso, considere la posibilidad de utilizar una arquitectura de múltiples cuentas para trabajar con las cuotas de QLDB. Para obtener más información, consulte aislamiento en silos de cuentas en el documento técnico AWS estrategias de aislamiento de inquilinos de SaaS.

Tamaño del documento

El tamaño máximo de un documento codificado en formato `IonBinary` es de 128 KB. No podemos proporcionar un límite exacto para el tamaño de un documento en formato `IonText` porque la conversión de texto a binario varía considerablemente en función de la estructura de cada documento. QLDB admite documentos con contenido abierto, por lo que cada estructura de documento única altera el cálculo del tamaño.

Tamaño de transacción

El tamaño máximo de una transacción en QLDB es de 4 MB. El tamaño de una transacción se calcula en función de la suma de los siguientes factores.

Deltas

Los cambios en el documento generados por todas las instrucciones de la transacción. En una transacción que afecta a varios documentos, el tamaño delta total es la suma del delta individual de cada documento afectado.

Metadatos

Los metadatos de la transacción generados por el sistema que están asociados a cada documento afectado.

Índices

Si se define un índice en una tabla que se ve afectada por la transacción, la entrada de índice asociada también genera un delta.

Historial

Como todas las revisiones de los documentos se conservan en QLDB, todas las transacciones también se anexan al historial.

Inserciones: cada documento insertado en una tabla también tiene una copia insertada en su tabla de historial. Por ejemplo, un documento de 100 KB recién insertado genera un mínimo de 200 KB de deltas en una transacción. (Se trata de una estimación aproximada que no incluye metadatos ni índices).

Actualizaciones: cualquier actualización de un documento, incluso de un solo campo, crea una nueva revisión de todo el documento en el historial, más o menos el delta de la actualización. Esto significa que una pequeña actualización en un documento grande seguiría generando un gran delta de transacciones. Por ejemplo, si se añaden 2 KB de datos a un documento existente de 100 KB, se crea una nueva revisión de 102 KB en el historial. Esto suma al menos 104 KB de deltas totales en una transacción. (De nuevo, se trata de una estimación que no incluye metadatos ni índices).

Eliminaciones: al igual que las actualizaciones, cualquier transacción de eliminación crea una nueva revisión del documento en el historial. Sin embargo, la revisión `DELETE` recién creada es más pequeña que el documento original porque tiene datos de usuario null y solo contiene metadatos.

Restricciones en la nomenclatura

En la siguiente tabla, se describen las restricciones en la nomenclatura de Amazon QLDB.

Nombre del libro mayor Nombre de la secuencia del diario	• Deben contener entre 1 y 32 caracteres alfanuméricos o guiones. • Debe tener una letra o un número para el primer y el último carácter. • No deben ser todos números. • No pueden contener dos guiones consecutivos. • Distingue entre mayúsculas y minúsculas.
Nombre de la tabla	• Las etiquetas solo deben contener de 1 a 128 caracteres alfanuméricos o guiones bajos. • Debe tener una letra o un guion bajo para el primer carácter. • Puede contener cualquier combinación de caracteres alfanuméricos y guiones bajos para el resto de los caracteres. • Distingue entre mayúsculas y minúsculas. • No debe ser una palabra reservada para QLDB PartiQL.

Información relacionada con Amazon QLDB

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con este servicio.

Temas

Documentación técnica

- Preguntas frecuentes sobre Amazon QLDB: preguntas frecuentes sobre el producto.
- Precios de Amazon QLDB: información y ejemplos de precios de AWS.
- AWS re:Post: foro comunitario de preguntas y respuestas de AWS.
- Amazon Ion: guías para desarrolladores, guías para usuarios y referencias sobre el formato de datos de Amazon Ion.
- PartiQL: documento de especificaciones y tutoriales generales para el lenguaje de consultas PartiQL.
- Talleres de QLDB: talleres que ofrecen ejemplos prácticos del uso de Amazon QLDB para crear aplicaciones de sistemas de registro, incluidos los siguientes laboratorios:
  - Aprender los fundamentos de QLDB
  - Trabajar con Amazon Ion y convertir Ion a JSON (Java) y viceversa
  - Uso de AWS Glue y Amazon Athena para habilitar los datos de QLDB en un lago de datos
  - Transmisión de datos de QLDB a una instancia de base de datos de Amazon Aurora MySQL
- Proteja los datos de calidad con Amazon QLDB: una implementación de soluciones AWS que muestra cómo evitar que los atacantes manipulen datos de calidad mediante el uso de QLDB para mantener un historial preciso de los cambios en los datos. AWS Las implementaciones de soluciones lo ayudan a resolver problemas frecuentes y a construir más rápidamente con AWS.

Repositorios de GitHub

Controladores

- Controlador .NET: una implementación .NET del controlador de QLDB.
- Controlador Go: una implementación Go del controlador de QLDB.
- Controlador Java: una implementación Java del controlador de QLDB.
- Controlador Node.js: una implementación Node.js del controlador de QLDB.
- Controlador Python: una implementación Python del controlador de QLDB.

Intérprete de comandos

- Intérprete de comandos de QLDB: implementación en Python y Rust de una interfaz de línea de comandos para la API de datos transaccionales de QLDB.

Aplicaciones de muestra

- Aplicación Java DMV: aplicación tutorial que se basa en un caso de uso del Departamento de Vehículos Motorizados (DMV). Demuestra las operaciones básicas y las mejores prácticas para usar QLDB y el controlador QLDB para Java.
- Aplicación DMV para .NET: aplicación tutorial basada en DMV que muestra las operaciones básicas y las mejores prácticas para usar QLDB y el controlador QLDB para .NET.
- Aplicación DMV para Node.js: aplicación tutorial basada en DMV que muestra las operaciones básicas y las mejores prácticas para usar QLDB y el controlador QLDB para Node.js.
- Aplicación DMV para Python: aplicación tutorial basada en DMV que muestra las operaciones básicas y las mejores prácticas para usar QLDB y el controlador QLDB para Python.
- Programa de carga de libro mayor: un marco de Java para la carga asíncrona de datos en un libro mayor de QLDB a alta velocidad mediante un canal de entrega compatible (AWS DMS, Amazon SQS, Amazon SNS, Kinesis Data Streams, Amazon MSK o EventBridge).
- Procesador de exportación: un marco Java ampliable que gestiona el trabajo de procesar las exportaciones de QLDB en Amazon S3 leyendo el resultado de la exportación e iterando los bloques exportados en secuencia.
- Lambda de ejemplo de secuencias de QLDB en Python: aplicación que demuestra cómo consumir secuencias de QLDB mediante una función AWS Lambda para enviar datos de QLDB a un tema de Amazon SNS, que tiene una cola de Amazon SQS suscrita.

- Ejemplo de integración de OpenSearch para secuencias de QLDB: aplicación de Python que muestra cómo integrar Amazon OpenSearch Service con QLDB mediante secuencias.
- Aplicación de doble entrada: aplicación Java que muestra cómo modelar una aplicación de libro mayor financiero de doble entrada mediante QLDB.
- QLDB KVS para Node.js: biblioteca sencilla de interfaz de almacenamiento de valores clave para QLDB con funciones adicionales para la verificación de documentos.

Amazon Ion y PartiQL

- Bibliotecas Amazon Ion: bibliotecas, herramientas y documentación admitidas por el equipo Ion.
- Implementación de PartiQL: implementación, especificación y tutoriales de PartiQL.

Publicaciones y artículos de blog de AWS

- Cómo creó Earnin su servicio de libro mayor con Amazon QLDB (16 de febrero de 2023): describe cómo Earnin.com utilizó QLDB para crear un servicio de libro mayor que proporcionara a sus usuarios una aplicación financiera móvil moderna y con todas las funciones.
- Exporte y analice los datos del diario de Amazon QLDB con AWS Glue y Amazon Athena (19 de diciembre de 2022): analiza cómo puede utilizar la característica de exportación de QLDB con AWS Glue y Athena para proporcionar funciones de informes y análisis a sus arquitecturas basadas en libros mayores.
- Cómo Shinsegae International mejora la experiencia del cliente y evita la falsificación con Amazon QLDB (3 de agosto de 2022): describe cómo Shinsegae International creó un servicio de verificación de autenticidad digital con Amazon QLDB para informar a los clientes sobre la autenticidad de los artículos de lujo y proporcionar el historial de compras y distribución de los productos.
- BungKusit utiliza Amazon QLDB y la tecnología ISV de VeriDoc Global para mejorar la experiencia de los clientes y los agentes de entrega (29 de abril de 2022). Muestra cómo una empresa de logística, BungKusit, utilizó QLDB para implementar una plataforma centralizada y transparente para la comunicación interindustrial con un registro de transacciones inmutable y verificable criptográficamente.
- Utilice Amazon QLDB como un almacén de valores clave inmutable con una API de REST y JSON (14 de febrero de 2022): presenta una forma de trabajar con QLDB como un almacén de valores clave inmutable y cómo utilizar sus características de auditoría a través de una API de REST.

- Creación de un sistema bancario central con Amazon QLDB (21 de enero de 2022): muestra cómo utilizar QLDB para crear un sistema de contabilidad bancaria básico. También muestra por qué QLDB es una base de datos adecuada para su propósito y que se adapta a las necesidades del sector de los servicios financieros.
- Cómo Specright usa Amazon QLDB para crear una red de cadena de suministro rastreable (21 de enero de 2022): muestra cómo Specright usa QLDB para crear una red de cadena de suministro rastreable que permite a las marcas mayoristas, minoristas y fabricantes compartir datos críticos de la cadena de suministro y los datos de especificaciones de empaque.
- Cómo proporciona fEMR datos médicos criptográficamente seguros y verificables con Amazon QLDB (23 de diciembre de 2021): analiza cómo el equipo de fEMR utilizó QLDB y otros servicios gestionados por AWS para facilitar sus proyectos de ayuda.
- Supervise los patrones de acceso a las consultas de Amazon QLDB (8 de noviembre de 2021): muestra cómo asociar las transacciones de QLDB y las instrucciones PartiQL que se ejecutaron en las transacciones con las solicitudes de API recibidas a través de Amazon API Gateway.
- Cree una operación CRUD y un flujo de datos sencillos en Amazon QLDB mediante AWS Lambda (28 de septiembre de 2021): muestra cómo realizar operaciones CRUD (crear, leer, actualizar y eliminar, por sus siglas en inglés) en QLDB mediante funciones AWS Lambda.
- Verificación criptográfica en la vida real con Amazon QLDB (26 de agosto de 2021): analiza el valor de la verificación criptográfica en un libro mayor de QLDB en el contexto de un caso de uso real.
- Verifique las condiciones de entrega con Accord Project y Amazon QLDB: parte 1, parte 2 (28 de junio de 2021): analiza cómo aplicar la tecnología de contratos legales inteligentes para verificar las condiciones de entrega con los recursos Accord Project y QLDB de código abierto.
- Flujo de datos de Amazon QLDB mediante AWS CDK (7 de junio de 2021): muestra cómo utilizar AWS Cloud Development Kit (AWS CDK) para configurar QLDB, rellenar los datos de QLDB mediante funciones AWS Lambda y configurar la secuencia de QLDB para proporcionar resiliencia a los datos del libro mayor.
- Demostración del control de acceso detallado en QLDB (1 de junio de 2021): muestra cómo comenzar a utilizar los permisos detallados AWS Identity and Access Management (IAM) para un libro mayor de QLDB.
- Procesamiento de secuencias con secuencias de DynamoDB y de QLDB (23 de noviembre de 2020): proporciona una breve comparación entre las secuencias de DynamoDB y las secuencias de QLDB, así como consejos para empezar a usarlas.

- Transmisión de datos desde Amazon QLDB a OpenSearch (19 de agosto de 2020): describe cómo transmitir datos desde QLDB a Amazon OpenSearch Service para permitir la búsqueda de texto enriquecido y los análisis posteriores, como la combinación o las métricas entre registros.
- Cómo transmití datos de Amazon QLDB a DynamoDB mediante Nodejs prácticamente en tiempo real (7 de julio de 2020): describe cómo transmitir datos de QLDB a DynamoDB para admitir una latencia de un solo dígito y consultas de valores clave con una escalabilidad infinita.
- Creación de una interfaz GraphQL para Amazon QLDB con AWS AppSync: parte 1, parte 2 (4 de mayo de 2020): analiza cómo integrar QLDB y AWS AppSync para proporcionar una API versátil con tecnología GraphQL sobre un libro mayor de QLDB.

Medios

Vídeos de AWS

- AWSTutoriales y demostraciones: secuencia de QLDB a Aurora (17 de marzo de 2023; 21 minutos): en este vídeo se explican los conceptos fundamentales para implementar la capacidad de transmisión de QLDB y se muestra cómo configurar el flujo de datos desde QLDB a una base de datos descendente de Amazon Aurora MySQL.
- ArcBlock: aprovechar Amazon QLDB para crear una solución de identidad descentralizada (31 de mayo de 2022; 4 minutos): ArcBlock explica la solución de identidad descentralizada que se creó mediante QLDB.
- AWS re:Invent 2020: uso de Amazon QLDB como base de datos de un sistema de confianza para las principales aplicaciones empresariales (5 de febrero de 2021; 30 minutos): descubra cómo los primeros usuarios de Amazon QLDB aplicaron las propiedades únicas de la base de datos de libro mayor para la procedencia de los datos y la verificabilidad criptográfica para implementar sistemas de registro con la integridad de los datos integrada.
- AWS re:Invent 2020: creación de una aplicación sin servidor con Amazon QLDB (5 de febrero de 2021; 28 minutos): aprenda a crear, probar y optimizar una aplicación sin servidor totalmente funcional combinando Amazon QLDB con servicios como AWS Lambda, Amazon Kinesis y Amazon DynamoDB.
- AWS re:Invent 2020: creación de aplicaciones basadas en auditorías que mantienen la integridad de los datos con Amazon QLDB (5 de febrero de 2021; 18 minutos): esta sesión analiza los problemas que Amazon QLDB puede resolver, responde a sus preguntas sobre cuándo y por qué utilizaría una base de datos de libro mayor y comparte casos de uso de clientes como Osano.

- Cómo almacenar registros inmutables de forma centralizada mediante Amazon QLDB con aplicaciones .NET (7 de diciembre de 2020; 10 minutos): demostración de cómo usar QLDB con aplicaciones .NET para almacenar registros inmutables de forma centralizada.
- Taller virtual: creación de sistemas de registro basados en libro mayor con QLDB y Java - Charlas técnicas AWS en línea (29 de julio de 2020; 87 minutos): taller dirigido por un instructor que recorre el laboratorio Working With Ion Immersion Day para crear un programa de carga de datos utilizando la biblioteca Amazon Ion y el controlador de QLDB para Java.
- Taller para desarrolladores: uso de la base de datos de libro mayor QLDB Inmutable de AWS en el nodo ABT (22 de junio de 2020; 34 minutos): una guía paso a paso sobre cómo establecer y configurar QLDB en el nodo ABT. También explica cómo instalar y configurar Blockchain Explorer y Boarding Gate Blocklets de ArcBlock para conectarse a QLDB.
- AWSCharla técnica: la perspectiva de un cliente sobre la creación de una aplicación de sistema de registro activada por eventos con Amazon QLDB (31 de marzo de 2020; 29 minutos): charla técnica de Matt Lewis, Data Hero AWS y arquitecto principal de la Agencia de Licencias de Conductores y Vehículos (DVLA, por sus siglas en inglés) del Reino Unido, en la que se explica el caso de uso de QLDB de la DVLA y la arquitectura basada en eventos de su aplicación.
- AWS re:Invent 2019: Por qué se necesita una base de datos de libro mayor: BMW, DVLA y Sage analizan los casos de uso (5 de diciembre de 2019; 47 minutos): una presentación a cargo de los clientes BMW, la organización gubernamental británica DVLA y Sage en la que se analizan las razones para utilizar una base de datos de libro mayor y se comparten sus casos de uso de QLDB.
- AWS re:Invent 2019: Amazon QLDB: análisis profundo de un ingeniero sobre por qué esto cambia las reglas del juego (5 de diciembre de 2019; 50 minutos): presentación de Andrew Certain (ingeniero distinguido de AWS) en la que se analiza la arquitectura única de QLDB, basada en el diario, junto con sus diversas innovaciones. Incluye hash criptográfico, árboles de Merkle, replicación de múltiples zonas de disponibilidad y compatibilidad con PartiQL.
- Creación de aplicaciones con Amazon QLDB, una base de datos de libro mayor única en su especie: charlas técnicas AWS en línea (19 de noviembre de 2019; 51 minutos): una charla técnica exhaustiva que describe las características únicas de QLDB y detalles sobre cómo utilizar las funciones principales. Incluye consultar el historial completo de sus datos, verificar criptográficamente los documentos y diseñar un modelo de datos.

Podcasts

- ¿Por qué los clientes eligen Amazon QLDB? (5 de julio de 2020; 33 minutos): un debate en el que se explica la definición de una base de datos de libro mayor, en qué se diferencia de una cadena de bloques y cómo la utilizan los clientes en la actualidad.

Recursos generales de AWS

- Clases y talleres: enlaces a cursos basados en roles y especializados, además de laboratorios autoguiados para ayudarlo a desarrollar sus conocimientos sobre AWS y obtener experiencia práctica.
- Centro para desarrolladores de AWS: explore los tutoriales, descargue herramientas y obtenga información sobre los eventos para desarrolladores de AWS.
- Herramientas para desarrolladores de AWS: enlaces a herramientas para desarrolladores, SDK, conjuntos de herramientas de IDE y herramientas de línea de comandos para desarrollar y administrar aplicaciones de AWS.
- Centro de recursos de introducción: aprenda a configurar su Cuenta de AWS, únase a la comunidad de AWS y lance su primera aplicación.
- Tutoriales prácticos: comience con tutoriales paso a paso antes de lanzar su primera aplicación en AWS.
- Documentos técnicos de AWS: enlaces a una lista completa de documentos técnicos de AWS que tratan una gran variedad de temas técnicos, como arquitecturas, seguridad y economía de la nube, escritos por arquitectos de soluciones de AWS o expertos técnicos.
- AWS SupportCentro de : punto para crear y administrar los casos de AWS Support. También incluye enlaces a otros recursos útiles como foros, preguntas técnicas frecuentes, estado de los servicios y AWS Trusted Advisor.
- AWS Support: la página web principal para obtener información acerca de AWS Support, un canal de soporte individualizado y de respuesta rápida que le ayudará a crear y ejecutar aplicaciones en la nube.
- Contacte con nosotros – Un punto central de contacto para las consultas relacionadas con la facturación AWS, cuentas, eventos, abuso y demás problemas.
- AWSTérminos del sitio de : información detallada sobre nuestros derechos de autor y marca comercial, su cuenta, licencia y acceso al sitio, entre otros temas.

Historial de versiones de Amazon QLDB

En la siguiente tabla se describen los cambios importantes en cada versión de Amazon QLDB y las actualizaciones correspondientes de la Guía para desarrolladores de Amazon QLDB. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a la fuente RSS.

- Versión de la API: 02-01-2019
- Última actualización de la documentación: 3 de enero de 2023

Cambio	Descripción	Fecha
Directrices IAM actualizadas	Guía actualizada para implementar las prácticas recomendadas de IAM. Para obtener más información, consulte prácticas recomendadas de seguridad en IAM.	3 de enero de 2023
Actualización a las políticas administradas de AWS	Amazon QLDB actualizó las políticas administradas de AWS existentes `AmazonQLDBFullAccess` y `AmazonQLDBConsoleFullAccess`. Estas políticas tienen un nuevo permiso que permite a las entidades principales editar las revisiones de los documentos mediante un procedimiento almacenado en PartiQL. Para obtener más información, consulte políticas administradas por AWS para Amazon QLDB.	4 de noviembre de 2022
Edición de datos	Amazon QLDB ahora admite el procedimiento almacenad	3 de noviembre de 2022

	o PartiQL de `REDACT_REVISION` para los libros mayores que se crearon a partir del 22 de julio de 2021. Con este procedimiento almacenado, puede eliminar permanentemente las revisiones de documentos inactivas del historial y, al mismo tiempo, mantener la integridad general de los datos de su libro mayor. Para obtener más información, consulte edición de revisiones de documentos.	
Controlador de Node.js v3	El controlador Amazon QLDB para la versión 3.0 de Node.js ya está disponible de forma general. Esta versión introduce el soporte para la versión 3 de AWS SDK for JavaScript. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-nodejs.	26 de septiembre de 2022
Controlador Go v3	El controlador Amazon QLDB para la versión 3.0 de Go ya está disponible de forma general. Esta versión introduce el soporte para la versión 2 de AWS SDK for Go. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-go.	11 de agosto de 2022

Nuevo editor de consultas PartiQL	Un nuevo editor de consultas PartiQL en la consola de Amazon QLDB ya está disponible de forma general. El nuevo editor PartiQL de QLDB proporciona una interfaz mejorada para crear consultas, depurar transacciones y explorar los resultados. Para obtener información sobre cómo abrir y usar el editor, consulte Acceso a Amazon QLDB mediante la consola.	22 de junio de 2022
Mapeador de objetos Ion para el controlador .NET	El controlador Amazon QLDB para .NET versión 1.3 introduce la compatibilidad con el mapeador de objetos Ion. Esta característica le permite evitar por completo la necesidad de realizar conversiones manuales entre los tipos de Amazon Ion y los tipos nativos de C#. Para ver el historial completo de cambios del mapeador de objetos Ion, consulte el archivo CHANGELOG.md en el repositorio de GitHub `amzn/ion-object-mapper-dotnet` .	19 de enero de 2022

Formato de exportación de diarios JSON	Amazon QLDB ahora admite el formato de salida de JSON Lines para la exportación de diarios. Para obtener más información, consulte exportación de datos de diarios desde Amazon QLDB.	21 de diciembre de 2021
Solución de problemas de Amazon QLDB	Se agregó un nuevo tema de resolución de problemas que proporciona orientación para una lista agregada de errores comunes que puede encontrar al usar Amazon QLDB.	8 de diciembre de 2021
Lanzamiento de nueva región	Amazon QLDB ya está disponible en la región Canadá (centro). Para ver una lista completa de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en Referenci a general de Amazon Web Services.	11 de noviembre de 2021
Prevención del suplente confuso entre servicios	Amazon QLDB ahora es compatible con el uso de las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos de IAM para evitar el problema del suplente confuso. Para obtener más información, consulte Prevención del suplente confuso entre servicios.	8 de noviembre de 2021

Intérprete de comandos de Amazon QLDB v2	La versión 2.0 del intérprete de comandos de Amazon QLDB, que está escrita en Rust, ya está disponible de forma general. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-shell.	14 de octubre de 2021
Actualización a las políticas administradas de AWS	Amazon QLDB actualizó las políticas administradas de AWS existentes `AmazonQLDBFullAccess` y `AmazonQLDBConsoleFullAccess` . Estas políticas tienen permisos recientemente agregados para permitir a las entidades principales transferir cualquier recurso de rol de IAM de su cuenta al servicio QLDB. Esto es necesario para todas las exportaciones de diarios y solicitudes de secuencias. Para obtener más información, consulte políticas administradas por AWS para Amazon QLDB.	2 de septiembre de 2021

Claves AWS KMS administradas por el cliente	Amazon QLDB ahora admite el cifrado en reposo mediante claves administradas por el cliente en AWS Key Management Service (AWS KMS) para nuevos recursos del libro mayor. Para obtener más información, consulte cifrado en reposo de Amazon QLDB.	22 de julio de 2021
Actualización de la política administrada de AWS	Amazon QLDB actualizó la política administrada `AmazonQLDBReadOnly` de AWS existente para eliminar una acción duplicada `qldb:GetBlock` que anteriormente aparecía en la lista dos veces. Para obtener más información, consulte políticas administradas por AWS para Amazon QLDB.	1 de julio de 2021
Lanzamiento de nueva región	Amazon QLDB ya se encuentra disponible en la región de Europa (Londres) . Para ver una lista completa de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de Amazon Web Services.	24 de junio de 2021

Actualización a las políticas administradas de AWS	Amazon QLDB actualizó las políticas administradas de AWS existentes `AmazonQLDBFullAccess` y `AmazonQLDBConsoleFullAccess` . A estas políticas se les han agregado permisos recientemente para permitir a las entidades principales actualizar el modo de permisos en todos los libros mayores y ejecutar todos los comandos PartiQL en todos los libros mayores con permisos `STANDARD`. Para obtener más información, consulte políticas administradas por AWS para Amazon QLDB.	27 de mayo de 2021
Modo de permisos estándar	Amazon QLDB ahora admite un modo de permisos `STANDARD` para los recursos del libro mayor. El modo de permisos estándar habilita el control de acceso con granularidad más precisa para libros mayores, tablas y comandos de PartiQL. Para obtener más información, consulte Introducción al modo de permisos estándar.	27 de mayo de 2021

Estadísticas de instrucciones PartiQL	La característica de estadísticas de instrucción PartiQL ya está disponible en el Editor de consultas de la consola de Amazon QLDB. Para obtener más información, consulte cómo obtener estadísticas de instrucciones.	24 de mayo de 2021
Tutorial: verificación de datos mediante un SDK de AWS	Se agregó un tutorial paso a paso con ejemplos de código que muestran cómo verificar un hash de revisión y un hash de bloque en Amazon QLDB mediante la API de QLDB a través de un SDK de AWS. Para obtener más información, consulte el Tutorial: Verificación de datos mediante un SDK de AWS.	6 de mayo de 2021
Controlador .NET v1.2	El controlador Amazon QLDB para la versión 1.2 de .NET ya está disponible de forma general. Esta versión presenta las API asíncronas. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-dotnet.	1 de abril de 2021
Instrucción `DROP INDEX` de PartiQL	PartiQL en Amazon QLDB ahora admite la instrucción DROP INDEX. Para obtener más información, consulte anular índices.	3 de marzo de 2021

Estadísticas de instrucciones PartiQL	La característica de estadísticas de instrucción PartiQL ya está disponible en la última versión del controlador Amazon QLDB para todos los lenguajes compatibles, incluidos .NET, Go y Python. Para obtener más información, consulte cómo obtener estadísticas de instrucciones.	25 de febrero de 2021
Tutorial de inicio rápido de TypeScript	Se agregaron ejemplos de código de TypeScript en el Tutorial de inicio rápido del controlador Amazon QLDB para Node.js.	28 de diciembre de 2020
Estadísticas de instrucciones PartiQL	La última versión del controlador Amazon QLDB para Java y Node.js ahora proporciona estadísticas de ejecución de instrucciones que pueden ayudarlo a ejecutar instrucciones PartiQL de forma más eficiente. Para obtener más información, consulte cómo obtener estadísticas de instrucciones.	22 de diciembre de 2020

Referencias a libros de recetas sobre controladores y tutoriales de inicio rápido	Se agregaron referencias de libros de recetas para los controladores Go y Node.js, tutoriales de inicio rápido para los controladores de Java y Python y una guía para trabajar con Amazon Ion en QLDB. Para obtener más información, consulte introducción al controlador Amazon QLDB.	24 de noviembre de 2020
Intérprete de comandos de Amazon QLDB v1.1	La versión 1.1 del intérprete de comandos de Amazon QLDB ya está disponible de forma general. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-shell.	26 de octubre de 2020
Controlador Amazon QLDB para Go	El controlador Amazon QLDB para Go ya está disponible de forma general. Este controlador es de código abierto en GitHub y le permite usar AWS SDK for Go para interactuar con la API de datos transaccionales de QLDB. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-go.	20 de octubre de 2020

Recomendaciones de configuración del controlador Node.js	Se agregó una nueva sección que proporciona recomendaciones de configuración para el controlador Amazon QLDB para Node.js, incluida la forma de reducir la latencia mediante la reutilización de las conexiones con keep-alive.	16 de octubre de 2020
Creación de índices en tablas no vacías	Amazon QLDB ahora admite la creación de índices en tablas no vacías. Para obtener más información, consulte administración de índices.	30 de septiembre de 2020
Optimización del rendimiento de las consultas	Se agregó una nueva sección que describe las restricciones de consulta en Amazon QLDB y proporciona orientación para optimizar el rendimiento de las consultas dadas estas restricciones.	18 de septiembre de 2020
Libro de recetas de referencia para el controlador de Java	Se agregó una referencia a un libro de recetas que muestra ejemplos de código de casos de uso comunes del controlador Amazon QLDB para Java.	3 de septiembre de 2020

Gestión de sesiones con el controlador	Se agregó una nueva sección que ofrece una descripción general de la administración de sesiones con el controlador en Amazon QLDB y describe cómo el controlador QLDB administra las sesiones cuando se ejecutan transacciones de datos.	1 de septiembre de 2020
Controlador Java v2.0	El controlador Amazon QLDB para la versión 2.0 de Java ya está disponible de forma general. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-java.	28 de agosto de 2020
Controlador Node.js v2.0	El controlador Amazon QLDB para la versión 2.0 de Node.js ya está disponible de forma general. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-nodejs.	27 de agosto de 2020
Información relacionada	Se agregó un tema nuevo que contiene enlaces a información relacionada y recursos adicionales para ayudarlo a comprender Amazon QLDB y trabajar con él.	24 de agosto de 2020

Controlador Python v3.0	El controlador Amazon QLDB para la versión 3.0 de Python ya está disponible de forma general. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-python.	20 de agosto de 2020
Controlador Amazon QLDB para Go	Ya está disponible una versión preliminar del controlador Amazon QLDB para Go. Este controlador le permite usar AWS SDK for Go para interactuar con la API de datos transaccionales de QLDB. Para obtener más informaci ón, consulte el Controlad or Amazon QLDB para Go (versión preliminar).	6 de agosto de 2020
Libro de recetas de referencia para el controlador de Python	Se agregó una referencia a un libro de recetas que muestra ejemplos de código de casos de uso comunes del controlador Amazon QLDB para Python.	24 de julio de 2020
Identificadores únicos en Amazon QLDB	Se agregó una nueva sección que describe las propiedad es y las pautas de uso de los identificadores únicos en Amazon QLDB.	9 de julio de 2020

Recurso AWS CloudFormation para secuencias de diarios	Amazon QLDB ahora admite un recurso para crear secuencias de diarios mediante una plantilla AWS CloudFormation. Para obtener más información, consulte el recurso AWS::QLDB::Stream en la Guía de usuario de AWS CloudFormation.	9 de julio de 2020
Libro de recetas de referencia para el controlador .NET	Se agregó una referencia a un libro de recetas que muestra ejemplos de código de casos de uso comunes del controlador Amazon QLDB para .NET.	July 1, 2020
Controlador Amazon QLDB para .NET	El controlador Amazon QLDB para .NET ya está disponible de forma general. Este controlador es de código abierto en GitHub y le permite usar AWS SDK for .NET para interactuar con la API de datos transaccionales de QLDB. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-dotnet.	26 de junio de 2020

Controlador Amazon QLDB para Node.js	El controlador Amazon QLDB para Node.js ya está disponible de forma general. Este controlador es de código abierto en GitHub y le permite usar SDK de AWS para JavaScript en Node.js para interactuar con la API de datos transaccionales de QLDB. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-nodejs.	5 de junio de 2020
Secuencias de diarios de Amazon QLDB	Amazon QLDB ahora permite crear una secuencia que capture todas las revisiones de documentos consignadas en su diario y entregue estos datos a Amazon Kinesis Data Streams en tiempo prácticamente real. Para obtener más información, consulte transmisión de datos de diarios de Amazon QLDB.	19 de mayo de 2020
Ejemplos de código de Amazon Ion	Se agregaron ejemplos de código que consultan y procesan datos de Amazon Ion en un libro mayor de Amazon QLDB mediante el controlador QLDB para Java, Node.js y Python. Para obtener más información, consulte ejemplos de código de Ion en Amazon QLDB.	12 de mayo de 2020

Modelo de concurrencia y contenido del diario	Se amplió el tema del modelo de concurrencia de Amazon QLDB para añadir información sobre el uso de índices para limitar los conflictos de control de concurrencia optimista (OCC). Se agregó una nueva sección que describe el contenido del diario en Amazon QLDB.	4 de mayo de 2020
Guía de inicio rápido para el controlador Node.js	Se agregó una guía de inicio rápido para el controlador Amazon QLDB para Node.js.	1 de mayo de 2020
Intérprete de comandos de Amazon QLDB	El intérprete de comandos de Amazon QLDB ya está disponible de forma general. Este intérprete de comandos es de código abierto en GitHub y proporciona una interfaz de línea de comandos que permite ejecutar instrucciones PartiQL en los datos del libro mayor. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-shell.	20 de abril de 2020
Certificaciones de conformidad	Amazon QLDB ahora cuenta con la certificación de los programas de conformidad AWS, incluidos la HIPAA y la ISO. Para obtener más información, consulte validación de la conformidad para Amazon QLDB.	3 de abril de 2020

Recomendaciones ampliadas para los controladores	Se amplió la sección de recomendaciones para los controladores de Amazon QLDB para que se aplique a todos los lenguajes de programación compatibles.	2 de abril de 2020
Intérprete de comandos de Amazon QLDB	Ya está disponible una versión preliminar del intérprete de comandos Amazon QLDB. Este intérprete de comandos proporciona una interfaz de línea de comandos que permite ejecutar instrucciones PartiQL en los datos del libro mayor. Para obtener más información, consulte acceder a Amazon QLDB mediante el intérprete de comandos de QLDB (solo API de datos) (versión preliminar).	23 de marzo de 2020
Controlador de Java v1.1	El controlador Amazon QLDB para la versión 1.1 de Java ya está disponible de forma general. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-java.	20 de marzo de 2020

Controlador Amazon QLDB para .NET	Amazon QLDB ahora ofrece una versión preliminar del controlador .NET. Este controlador le permite usar AWS SDK for .NET para interactuar con la API de datos transaccionales de QLDB. Para obtener más informaci ón, consulte el controlador Amazon QLDB para NET (versión preliminar).	13 de marzo de 2020
Controlador Amazon QLDB para Python	El controlador de Amazon QLDB para Python ya está disponible de forma general. Este controlador es de código abierto en GitHub y le permite usar AWS SDK for Python (Boto3) para interactuar con la API de datos transacci onales de QLDB. Para ver las notas de la versión, consulte el repositorio de GitHub awslabs/amazon-qldb-driver-python.	11 de marzo de 2020

Instrucción `UNDROP TABLE` de PartiQL y XML anidado	PartiQL en Amazon QLDB ahora admite instrucciones en lenguaje de manipulación de datos (DML) en las que las colecciones anidadas se especifican como origen en la cláusula `FROM`. Para obtener más información, consulte la instrucción FROM en la referencia de PartiQL. PartiQL de QLDB también admite la instrucción UNDROP TABLE. Para obtener más información, consulte eliminar tablas.	26 de febrero de 2020
Tema de introducción ampliado	Se amplió el tema de la introducción ¿Qué es Amazon QLDB? para incluir las nuevas secciones Descripción general de Amazon QLDB y De relacional a libro mayor.	24 de enero de 2020
Referencia PartiQL para las funciones compatibles	Se amplió la referencia PartiQL de Amazon QLDB para incluir información de uso detallada sobre las funciones de SQL compatibles. Para obtener más información, consulte funciones PartiQL.	2 de enero de 2020

Recomendaciones de controladores y errores comunes	Se agregaron nuevas secciones que describen las recomendaciones de controladores para el controlador Amazon QLDB para Java y los errores comunes para todos los lenguajes de controladores.	2 de enero de 2020
Lanzamiento de nuevas regiones	Amazon QLDB está ahora disponible en las regiones Asia-Pacífico (Seúl), Asia Pacífico (Singapur), Asia Pacífico (Sídney) y Europa (Fráncfort). Para ver una lista completa de las regiones disponibles, consulte Puntos de conexión y cuotas de Amazon QLDB en Referencia general de Amazon Web Services.	19 de noviembre de 2019
Controlador Amazon QLDB para Node.js	Amazon QLDB ahora ofrece una versión preliminar del controlador Node.js. Este controlador le permite usar SDK de AWS para JavaScript en Node.js para interactuar con la API de datos transaccionales de QLDB. Para obtener más información, consulte el controlador Amazon QLDB para Node.js (versión preliminar).	13 de noviembre de 2019

Controlador Amazon QLDB para Python	Amazon QLDB ahora ofrece una versión preliminar del controlador Python. Este controlador le permite usar AWS SDK for Python (Boto3) para interactuar con la API de datos transaccionales de QLDB. Para obtener más información, consulte el controlador Amazon QLDB para Python (versión preliminar).	29 de octubre de 2019
Presentación pública	Esta es la versión pública inicial de Amazon QLDB. Esta versión incluye la Guía para desarrolladores y la Referencia de la API para administración del libro mayor integrada.	10 de septiembre de 2019

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.