Universidad Francisco Marroquin

Facultad de Ciencias Economicas

Computer Science

Datos I

Sophia Gamarro: 20170356

MySql

Un candado es una bandera asociada a una tabla. MySQL permite que una sesión de cliente adquiera explícitamente un bloqueo de tabla para evitar que otras sesiones accedan a la misma tabla durante un período específico.

Una sesión de cliente puede adquirir o liberar bloqueos de tabla solo para sí misma. Y una sesión de cliente no puede adquirir ni liberar bloqueos de tabla para otras sesiones de cliente.

MySQL LOCK TABLES statement

La siguiente declaración adquiere explícitamente un bloqueo de tabla:

**LOCK** **TABLES** table\_name [**READ** | WRITE]

En esta sintaxis, especifica el nombre de la tabla que desea bloquear después de las palabras clave LOCK TABLES. Además, especifica el tipo de bloqueo, ya sea READ o WRITE.

MySQL le permite bloquear varias tablas especificando una lista de nombres de tablas separados por comas con tipos de bloqueo que desea bloquear después de las palabras clave.

Read Locks

Un candado READ tiene las siguientes características: Varias sesiones al mismo tiempo pueden adquirir un bloqueo READ para una tabla. Además, otras sesiones pueden leer datos de la tabla sin adquirir el bloqueo. La sesión que mantiene el bloqueo READ solo puede leer datos de la tabla, pero no puede escribir. Y otras sesiones no pueden escribir datos en la tabla hasta que se libere el bloqueo READ. Las operaciones de escritura de otra sesión se pondrán en estado de espera hasta que se libere el bloqueo READ. Si la sesión finaliza, ya sea normal o anormalmente, MySQL liberará todos los bloqueos implícitamente. Esta función también es relevante para el bloqueo de WRITE.

Write Locks

Un candado WRITE tiene las siguientes características:

• La única sesión que mantiene el candado de una tabla puede leer y escribir datos de la tabla.

• Otras sesiones no pueden leer ni escribir datos en la tabla hasta que se libere el bloqueo de WRITE.

READ vs. Write Locks

Los bloqueos de lectura son bloqueos "compartidos" que impiden que se adquiera un bloqueo de escritura, pero no otros bloqueos de lectura.

Los bloqueos de escritura son bloqueos “exclusivos” que evitan cualquier otro bloqueo de cualquier tipo.

SQLServer

El locking es esencial para el procesamiento exitoso de transacciones de SQL Server y está diseñado para permitir que SQL Server funcione sin problemas en un entorno multiusuario. El bloqueo es la forma en que SQL Server administra la concurrencia de transacciones. Básicamente, los bloqueos son estructuras en memoria que tienen propietarios, tipos y el hash del recurso que debe proteger. Un bloqueo como estructura en memoria tiene un tamaño de 96 bytes.

Para comprender mejor el bloqueo en SQL Server, es importante comprender que el bloqueo está diseñado para garantizar la integridad de los datos en la base de datos, ya que obliga a todas las transacciones de SQL Server a pasar la prueba ACID.

La prueba ACID consta de 4 requisitos que cada transacción debe aprobar con éxito:

* Atomicidad: requiere que una transacción que involucra dos o más partes discretas de información debe comprometer todas las partes o ninguna.
* Consistencia: requiere que una transacción debe crear un estado válido de datos nuevos, o debe revertir todos los datos al estado que existía antes de que se ejecutara la transacción.
* Aislamiento: requiere que una transacción que aún se esté ejecutando y que aún no haya enviado todos los datos, debe permanecer aislada de todas las demás transacciones.
* Durabilidad: requiere que los datos comprometidos se almacenen utilizando un método que preserve todos los datos en el estado correcto y estén disponibles para un usuario, incluso en caso de falla.

Modos de bloqueo

* Exclusivo (X): El bloqueo exclusivo lo impondrá la transacción cuando quiera modificar los datos de la página o fila, que es en el caso de las sentencias DML DELETE, INSERT y UPDATE. Se puede imponer un bloqueo exclusivo a una página o fila solo si no se ha impuesto otro bloqueo compartido o exclusivo en el destino.
* Compartido (S): este tipo de bloqueo, cuando se impone, reservará una página o fila para que esté disponible solo para lectura, lo que significa que cualquier otra transacción no podrá modificar el registro bloqueado siempre que el bloqueo esté activo. Además, un bloqueo compartido permitirá operaciones de escritura, pero no se permitirán cambios DDL
* Actualizar (U): este bloqueo es similar a un bloqueo exclusivo, pero está diseñado para ser más flexible de alguna manera. Se puede imponer un bloqueo de actualización a un registro que ya tiene un bloqueo compartido. Es importante comprender que el bloqueo de actualización es asimétrico con respecto a los bloqueos compartidos. Si bien el bloqueo de actualización se puede imponer en un registro que tiene el bloqueo compartido, el bloqueo compartido no se puede imponer en el registro que ya tiene el bloqueo de actualización.
* Intención (I): este bloqueo es un medio utilizado por una transacción para informar a otra transacción sobre su intención de adquirir un bloqueo. El propósito de dicho bloqueo es garantizar que la modificación de datos se ejecute correctamente evitando que otra transacción adquiera un bloqueo en el siguiente objeto de la jerarquía. Este es un tipo de bloqueo importante desde el punto de vista del rendimiento, ya que el motor de la base de datos de SQL Server inspeccionará los bloqueos de intención solo en el nivel de la tabla para verificar si es posible que la transacción adquiera un bloqueo de manera segura en esa tabla y, por lo tanto, el bloqueo de intención elimina necesita inspeccionar cada bloqueo de fila / página en una tabla para asegurarse de que la transacción pueda adquirir un bloqueo en toda la tabla
* Esquema (Sch)
* Actualización masiva (BU)

Regular Intent Locks

* Intención exclusiva (IX): cuando se adquiere un bloqueo exclusivo de intención (IX), indica a SQL Server que la transacción tiene la intención de modificar algunos de los recursos de jerarquía inferior adquiriendo bloqueos exclusivos (X) individualmente en esos recursos de jerarquía inferior
* Intención compartida (IS): cuando se adquiere un bloqueo de intención compartida (IS), indica a SQL Server que la transacción tiene la intención de leer algunos recursos de jerarquía inferior adquiriendo bloqueos compartidos (S) individualmente en los recursos inferiores de la jerarquía.
* Actualización de intención (IU): cuando se adquiere un bloqueo compartido de intención (IS), indica a SQL Server que la transacción tiene la intención de leer algunos de los recursos de jerarquía inferior adquiriendo bloqueos compartidos (S) individualmente en los recursos inferiores en la jerarquía.

Conversion Locks

* Compartido con intención exclusiva (SIX): este bloqueo indica que la transacción tiene la intención de leer todos los recursos en una jerarquía inferior y, por lo tanto, adquirir el bloqueo compartido en todos los recursos que son inferiores en la jerarquía y, a su vez, modificar parte de esos, pero no todos. En la práctica, esto significa que una vez que la transacción adquiere un bloqueo SIX en la tabla, adquirirá un bloqueo exclusivo de intención (IX) en las páginas modificadas y un bloqueo exclusivo (X) en las filas modificadas.
* Compartido con actualización de intenciones (SIU es una combinación de los bloqueos compartidos (S) y de actualización de intenciones (IU). Un ejemplo típico de este bloqueo es cuando una transacción utiliza una consulta ejecutada con la sugerencia y consulta PAGELOCK, luego la consulta de actualización. Después de que la transacción adquiere un bloqueo SIU en la tabla, la consulta con la sugerencia PAGELOCK adquirirá el bloqueo compartido (S) mientras que la consulta de actualización adquirirá el bloqueo de actualización de intención (IU)
* Actualizar con intención exclusiva (UIX): cuando los bloqueos de actualización (U) y los bloqueos exclusivos de intención (IX) se adquieren en recursos de jerarquía inferior en la tabla simultáneamente, la actualización con bloqueo exclusivo de intención se adquirirá en el nivel de la tabla como consecuencia
* Bloqueos de esquema (Sch): el motor de base de datos de SQL Server reconoce dos tipos de bloqueos de esquema: bloqueo de modificación de esquema (Sch-M) y bloqueo de estabilidad de esquema (Sch-S)
* Se adquirirá un bloqueo de modificación de esquema (Sch-M) cuando se ejecute una instrucción DDL, y evitará el acceso a los datos del objeto bloqueado a medida que se cambia la estructura del objeto.
  + Se adquirirá un bloqueo de estabilidad de esquema (Sch-S) mientras se compila y ejecuta una consulta dependiente del esquema y se genera el plan de ejecución. Este bloqueo en particular no bloqueará otras transacciones para acceder a los datos del objeto y es compatible con todos los modos de bloqueo excepto con el bloqueo de modificación de esquema (Sch-M).

Bloqueos de actualización masiva (BU): este bloqueo está diseñado para ser utilizado por operaciones de importación masiva cuando se emite con un argumento / sugerencia TABLOCK.

Cassandra

Cassandra no usa transacciones RDBMS ACID con mecanismos de retroceso o bloqueo, sino que ofrece transacciones atómicas, aisladas y duraderas con una consistencia eventual / ajustable que le permite al usuario decidir qué tan fuerte o eventual quiere que sea la consistencia de cada transacción.

LIGHTWEIGHT TRANSACTIONS

Si bien las transacciones duraderas con consistencia eventual / ajustable son bastante satisfactorias para muchos casos de uso, surgen situaciones en las que se necesita más. Las transacciones ligeras, también conocidas como comparar y establecer, que utilizan una coherencia linealizable probablemente puedan satisfacer esas necesidades.

Por ejemplo, si un usuario quiere asegurarse de que un inserto que está a punto de hacer en una tabla de cuentas nuevas sea único para un nuevo cliente, usaría la cláusula IF NOT EXISTS:

INSERT INTO customer\_account (customerID, customer\_email)

VALues ("LauraS", "lauras@gmail.com")

IF NOT EXISTS;

Cassandra está realizando cuatro viajes de ida y vuelta entre un nodo que propone una transacción ligera y las réplicas necesarias en el clúster para garantizar una ejecución adecuada y que el rendimiento se vea afectado. En consecuencia, reservar transacciones ligeras para aquellas situaciones en las que sean absolutamente necesarias; La consistencia eventual normal de Cassandra se puede utilizar para todo lo demás.

Un nivel de coherencia SERIAL permite leer el estado actual (y posiblemente no comprometido) de los datos sin proponer una nueva adición o actualización. Si una lectura SERIAL encuentra una transacción no confirmada en curso, la confirmará como parte de la lectura.

Atomicidad

En Cassandra, una escritura es atómica a nivel de fila, lo que significa que insertar o actualizar columnas en una fila se trata como una operación de escritura. Cassandra no admite transacciones en el sentido de agrupar actualizaciones de varias filas en una operación de todo o nada. Tampoco se revierte cuando una escritura se realiza correctamente en una réplica, pero falla en otras réplicas

Durabilidad

Los escritos en Cassandra son duraderos. Todas las escrituras en un nodo de réplica se registran tanto en la memoria como en un registro de confirmación en el disco antes de que se reconozcan como exitosas. Si ocurre una falla o falla del servidor antes de que las tablas de memoria se vacíen en el disco, el registro de confirmación se reproduce al reiniciar para recuperar las escrituras perdidas. Además de la durabilidad local (los datos se escriben inmediatamente en el disco), la replicación de datos en otros nodos refuerza la durabilidad.

DynamoDB

El cliente de bloqueo de DynamoDB es una biblioteca de Java ampliamente utilizada dentro de Amazon, que le permite resolver problemas de computación distribuida como elección de líder y bloqueo distribuido con código solo de cliente y una tabla de DynamoDB.

DynamoDB admite mecanismos, como escrituras condicionales, que son necesarios para bloqueos distribuidos. Sin embargo, el AWS SDK no incluye la lógica necesaria para implementar bloqueos distribuidos. El cliente de bloqueo de DynamoDB incluye la lógica de cliente necesaria para los bloqueos de aviso distribuidos en una interfaz de cliente fácil de usar. El protocolo de la biblioteca Java del cliente de bloqueo es de amplia aplicación y le recomendamos que lo aplique en otros idiomas.

El cliente de bloqueo de DynamoDB implementa un protocolo que permite que aplicaciones similares realicen bloqueos de aviso en cualquier parte del dominio de su problema, ya sea grande o pequeño. Este protocolo garantiza que sus jugadores "permanezcan en posesión del balón" durante un cierto período de tiempo.

**El protocolo de bloqueo**

Para un nuevo bloqueo, el cliente de bloqueo almacena un elemento de bloqueo en la tabla de bloqueo. Con el elemento, almacena el nombre de host del propietario, la duración de la concesión en milisegundos, un UUID único para el host y la hora del reloj del sistema host cuando se creó inicialmente el bloqueo. La tabla de bloqueo se parece a esto en la Consola de administración de AWS.

El siguiente diagrama de secuencia y arquitectura muestra el protocolo de bloqueo. La arquitectura del protocolo incluye un grupo EC2 Auto Scaling. El grupo abarca dos subredes en dos zonas de disponibilidad de una única región de AWS y dos instancias EC2 con una aplicación Java ejecutándose en cada subred. Cada una de las instancias está tratando de adquirir un **bloqueo en Moe.*Graphical user interface

Description automatically generated with medium confidence***

**DDL**

CREATE TABLE Cliente(

id\_cliente varchar(20),

email varchar(50),

address varchar (100),

ate\_created date,

phone\_number number,

loan\_limit number)

)

CREATE OR REPLACE CREAR\_CLIENTE(

id\_cliente varchar(20),

email varchar(50),

address varchar (100),

ate\_created date,

phone\_number number,

loan\_limit number)

iS

BEGIN

INSERT INTO Cliente ("id\_cliente", "name", "email", "address", "date\_created", "phone\_number", "loan\_limit")

VALUES (id\_cliente, name, email, address, date\_created, phone\_number, loan\_limit)

COMMIT;

END;

**Bibliografía:**

**Sasha. S (15 agosto 2017). Building Distributed Locks with the DynamoDB Lock Client. Obtenido de:** [**https://aws.amazon.com/blogs/database/building-distributed-locks-with-the-dynamodb-lock-client/**](https://aws.amazon.com/blogs/database/building-distributed-locks-with-the-dynamodb-lock-client/)

**Mysql Documentation. Obtenido de:** [**https://www.mysqltutorial.org/mysql-table-locking/#:~:text=A%20lock%20is%20a%20flag,table%20locks%20only%20for%20itself**](https://www.mysqltutorial.org/mysql-table-locking/#:~:text=A%20lock%20is%20a%20flag,table%20locks%20only%20for%20itself)**.**

**Nikola D. (16 de junio 2017). All about locking in SQL Server. Obtenido de: https://www.sqlshack.com/locking-sql-server/**