Gestión de Transacciones

M.C. Carlos Rojas Sánchez[1]

[1]Licenciatura en Informática
Universidad del Mar - Puerto Escodido

[Bases de Datos II]

1 Transacciones en MySQL

Transacciones en MySQL

- Una transacción tiene dos finales posibles, COMMIT y ROLLBACK. Por defecto, MySQL trae activado el modo autocommit, es decir, realizada una transacción (por ejemplo un INSERT, UPDATE o DELETE) el mismo es confirmado apenas es ejecutado.
- Se puede desactivar el autocommit ejecutando el comando: SET AUTOCOMMIT=0;
- Una vez deshabilitado el autocommit, tendremos que utilizar obligatoriamente el COMMIT para confirmar o ROLLBACK para deshacer la transacción.

Transacciones en MySQL

- Si se quiere deshabilitar el autocommit para una serie de comandos, lo ideal es utilizar START TRANSACTION.
- Al ejecutar una transacción, el motor de base de datos nos garantizará la atomicidad, consistencia, aislamiento y durabilidad (ACID) de la transacción (o conjunto de comandos) que se utilice.

Transacciones en MySQL

Ejemplo

- mysql> create table departamentos (codigo int not null default 0, nombre varchar (100), presupuesto int default null, primary key (codigo)) engine=Innodb;
- mysql> create table departamentos_externos (codigo int not null default 0, nombre varchar (100), presupuesto int default null, primary key (codigo)) engine=Innodb;

Transacciones en MySQL

Ejemplo

- Ahora, insertaremos registros de la tabla departamentos_externos a departamentos mediante una transacción:
  - mysql> start transaction;
  - mysql> select @A:=presupuesto FROM departamentos_externos WHERE codigo= 10; insert into departamentos (codigo, nombre, presupuesto)values (10,'Departamento test ',@A);

Transacciones en MySQL

Ejercicio 1

- START TRANSACTION;
- SELECT @A := presupuesto, @B := codigo, @C := nombre FROM departamentos_externos WHERE codigo=33; INSERT INTO departamentos(codigodep, nombredep, presupuesto) VALUES (@B , @C , @A);
- COMMIT ;

Transacciones en MySQL

Ejercicio 2

- START TRANSACTION;
- SELECT @A:=PRESUPUESTO FROM departamentos_externos WHERE codigo=11; UPDATE departamentos SET PRESUPUESTO = PRESUPUESTO + @A WHERE codigo=33;
- COMMIT;

Transacciones en MySQL

Ejercicio 3

- Realizar una transacción DELETE
- Qué sucede si al final de la transacción se pone ROLLBACK

2 Concepto y propiedades

Concepto y propiedades

- El sistema de base de datos normalmente está siendo accedido simultaneamente por muchos usuarios para hacer consultas como actualizaciones
- Hemos supuesto que ni el software ni el hardware pueden fallar en el intertanto de una operación

Motores de Almacenamiento

Concepto

Una transacción es una interacción con una estructura de datos compleja, compuesta por varios procesos que se han de aplicar uno después del otro. La transacción debe realizarse de una sola vez y sin que la estructura a medio manipular pueda ser alcanzada por el resto del sistema hasta que se hayan finalizado todos sus procesos

Tipos

- Transacciones de recuperación se accede a los datos para visualizarlos en la pantalla a modo de informe.
- Transacciones de actualización se insertan, borran o actualizan datos de la base de datos.
- Transacciones mixtas se mezclan operaciones de recuperación de datos y de actualización.

Propiedades

- Atomicidad (Atomicity): es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia (Consistency): es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

Propiedades

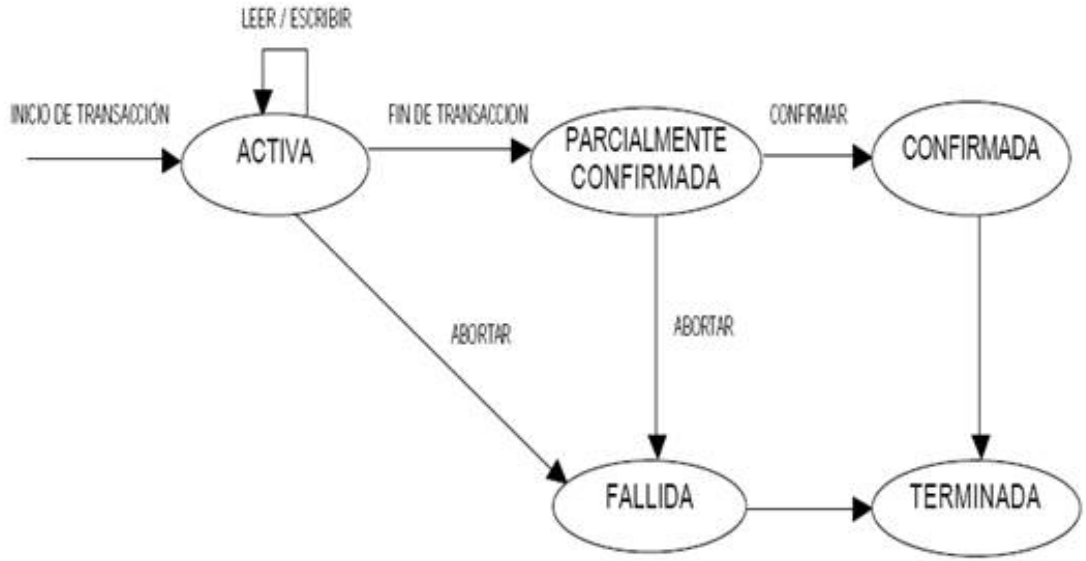
- Aislamiento (Isolation): es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información nunca generará ningún tipo de error.
- Permanencia (Durability): es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

3 Operaciones y estados

Estados

- Activa (Active): el estado inicial; la transacción permanece en este estado durante su ejecución.
- Parcialmente comprometida (Uncommited): Después de ejecutarse la ultima transacción.
- Fallida (Failed): tras descubrir que no se puede continuar la ejecución normal.
- Abortada (Rolled Back): después de haber retrocedido la transacción y restablecido la base de datos a su estado anterior al comienzo de la transacción.
- Comprometida (Commited): tras completarse con éxito.

Diagrama de estados



4 Optimización de consultas

Consejo 1

En una consulta SELECT, hay que evitar en lo posible el uso del comodín “*”, e indicar sólo los campos que se necesitan. Eso reducirá el tamaño de la consulta. Considerar que cada campo extra genera tiempo extra.

Consejo 2

- Usa frecuentemente el LIMIT
  - Usa LIMIT 1 Cuando sólo quieras una única fila; en estos casos, añadir LIMIT 1 a tu query puede mejorar significativamente la velocidad.
  - Si estamos seguros de que vamos a borrar un registro de la tabla, entonces podemos añadirle un LIMIT 1 al final y así nos aseguraremos que no se borrará en ninguno de los casos nada fuera de control. Esto es aplicable también para SELECTs y UPDATEs.
  - Cuando la cantidad de resultados a devolver por MySQL no va a ser utilizada en su conjunto, como cuando se usa un paginador, devolver la cantidad apropiada utilizando la cláusula LIMIT.

Consejo 3

Pruebe sus consultas con anterioridad, mediante el comando EXPLAIN. Con EXPLAIN podemos obtener toda la información sobre el modo en el que una consulta SQL se ejecutaría en el servidor. Es extremadamente útil para conocer la configuración de índices en las tablas, los índices que podrían ser configurados para mejorar su rendimiento, el número de filas que se revisan, el tipo de query, etc.

- EXPLAIN SELECT * FROM table;

Consejo 4

- Optimizar la cláusula WHERE:
  - Evite el uso de paréntesis innecesarios.
  - Use COUNT (*) sólo en consultas sin cláusula WHERE y que afecten a una única tabla.
  - Si conoce que el resultado de una claúsula GROUP BY o DISTINCT va a ser muy reducido haga uso de la opción SQL_SMALL_RESULT. MySQL usará tablas temporales de acceso rápido para el resultado en vez métodos de ordenación.

Consejo 5

Utilizar INNER JOIN, LEFT JOIN, RIGHT JOIN, para unir las tablas en lugar del WHERE, esto permite que a medida que se declaran las tablas se vayan uniendo mientras que si utilizamos el WHERE el motor genera primero el producto cartesiano de todos los registros de las tablas para luego filtrar las correctas, un trabajo definitivamente lento.

Consejo 6

No uses ORDER BY RAND(); el problema es que MySQL tendrá que ejecutar RAND() (que requiere de potencia de procesado) para cada una de las filas antes de ordenarlas y devolver una simple fila.

Consejo 7

Usa NOT NULL si puedes; las columnas NULL requieren espacio adicional en la fila a grabar donde los valores son NULL. Para las tablas MyISAM, cada columna NULL toma un bit extra, redondeando hacia arriba al byte más cercano.

5 Control de concurrencia

Modos de bloqueo InnoDB

Bloqueo a nivel de fila

- Compartido (Shared) (S) le permite a una transacción leer una fila
- Exclusivo (Exclusive) (X) le permite a una transacción actualizar o eliminar una fila

Bloqueo exclusivo

- Si una transacción A pone un bloqueo exclusivo (X) sobre una tupla t, entonces una solicitud de otra transacción B para establecer un bloqueo de cualquier tipo sobre t no puede ser atendida inmediatamente. En lugar de eso, la transacción B debe esperar a que la transacción A libere el bloqueo en la tupla t.

Bloqueo compartido

- Si la transacción A sostiene un bloqueo compartido (S) sobre una tupla t, entonces
  - Una solicitud de otra transacción B para un bloqueo X sobre t no puede ser atendido inmediatamente
  - Una solicitud de otra transacción B para un bloqueo S sobre t puede ser atendido inmediatamente. En consecuencia, tanto A como B sostendrán un bloqueo S sobre t

Más bloqueos

- InnoDB soporta bloqueo de granularidad múltiple, el cual permite que existan simultáneamente bloqueos en registros y bloqueos en tablas enteras. Para hacer práctico el nivel de bloqueo de granularidad múltiple, se emplean tipos adicionales de bloqueo, llamados bloqueos de intención. Los bloqueos de intención son bloqueos de tabla en InnoDB.

Bloqueos de intención

Asumiendo que la transacción T ha solicitado un bloqueo del tipo indicado en la tabla R:

- Intención compartida (Intention shared) (IS): La transacción T trata de establecer bloqueos S en tuplas individuales de la tabla T
- Intención exclusiva (Intention exclusive) (IX): La transacción T trata de establecer bloqueos X en las tuplas

Concurrencia

La ejecución concurrente de transacciones mejora la productividad y la utilización del sistema, y también reduce el tiempo de espera de las transacciones.

6 Comprobación de la seriabilidad y recuperabilidad

Seriabilidad

- Una planificación captura las acciones clave de las transacciones que afectan a la ejecución concurrente, tales como las operaciones leer y escribir, a la vez que se abstraen los detalles internos de la ejecución de la transacción.
- Puesto que las transacciones son programas, es difícil calcular cuáles son las operaciones exactas que realiza una transacción y cómo interaccionan las operaciones de varias transacciones.

Comprobación de la seriabilidad

Cuando se diseñan esquemas de control de concurrencia hay que demostrar que las planificaciones que genera el esquema son seriables

Comprobación de la seriabilidad

Considérese una planificación P. Se construye un grafo dirigido, llamado grafo de precedencia para P. Este grafo consiste en un par G = (V,A), siendo V un conjunto de vértices y A un conjunto de arcos.

Comprobación de la seriabilidad

El conjunto de vértices consiste en todas las transacciones que participan en la planificación. El conjunto de arcos consiste en todos los arcos $T_i \rightarrow T_j$ para los cuales se dan una de las tres condiciones siguientes

- T_i ejecuta escribir(Q) antes de que T_j ejecute leer(Q).
- T_i ejecuta leer(Q) antes de que T_j ejecute escribir(Q).
- T_i ejecuta escribir(Q) antes de que T_j ejecute escribir(Q).

Si existe un arco $T_i \rightarrow T_j$ en el grafo de precedencia, entonces en toda planificación secuencial P' equivalente a P, T_i debe aparecer antes de T_j.

Niveles de Aislamiento

Bibliografía I

- Jorge Sánchez.
  Principios sobre Bases de Datos Relacionales.
  Creative-Commons, 2004.
- Abraham Silberschatz, Henry F. Korth and S. Sudarshan.
  Fundamentos de Bases de Datos.
  McGRAW-HILL, 2002.