Transacciones

Transacciones en MySQL

Procedimientos Almacenados

Un procedimiento almacenado es un conjunto de instrucciones, que se almacenan bajo un **nombre**

Ventajas:

- Concentran lógica de negocio.
- Evitan acceso directo a las tablas.
- Reducen el tráfico de red.

Desventajas:

- Se aplican lógicas de negocios, pero no muy complejas.
- Difíciles de depurar.
- Ejecución de un procedimiento almacenado: Se invoca por su nombre
- Se pueden referenciar tablas, vistas, funciones y procedimientos almacenados.
- Se pueden utilizar instrucciones DML no DDL.

Transacciones

Una transacción en un Sistema de Gestión de Bases de Datos es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

- Integridad de los datos: transacciones no pueden finalizar en un estado intermedio.

Transacciones

Una transacción en un Sistema de Gestión de Bases de Datos es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

Integridad de los datos

Una transacción debe contar con ACID (un acrónimo inglés) que quiere decir: Atomicidad, consistencia, aislamiento y durabilidad.

InnoDB es totalmente compatible con ACID.

Atómicas. Esto significa que todas las operaciones tienen éxito o fallan.

Consistencia. Mantener el estado de la base de datos en un estado válido.

Aislamiento. Es el requisito de que otras operaciones no puedan acceder a los datos que se han modificado durante una transacción que aún no se ha completado.

Durabilidad. Es la capacidad del sistema de base de datos para poder recuperarse en caso que se presenten transacciones comprometidas contra cualquier tipo de falla del sistema.

Niveles de aislamiento

Problemas

- Dirty reads (Lecturas sucias): Supone que las transacciones en curso puedan leer el resultado de otras transacciones aún no confirmadas.
- Non-Repeatable reads (Lecturas no repetibles):Ocurre cuando una transacción activa vuelve a leer un dato cuyo valor difiere con respecto al de la anterior lectura.
- Phantom reads (Lecturas fantasma): una transacción en un momento lanza una consulta de selección con una condición y recibe en ese momento **N** filas y posteriormente vuelve a lanzar la misma consulta junto con la misma condición y recibe **M** filas con **M > N**.

Niveles de aislamiento

Read uncommitted (Lectura sin confirmación): En la práctica casi no se suele utilizar este nivel de aislamiento ya que es propenso a sufrir todos los problemas anteriormente descritos. En este nivel una transacción puede ver los resultados de transacciones aún no cometidas. Podemos apreciar que en este nivel no existe aislamiento alguno entre transacciones.

Read committed (Lectura confirmada): Es el predeterminado para la mayoría de gestores de bases de datos relacionales. Supone que dentro de una transacción únicamente se pueden ver los cambios de las transacciones ya cometidas. Soluciona el problema de las lecturas sucias, pero no el de las lecturas no repetibles ni tampoco el de las lecturas fantasmas.

Repeatable read (Lectura repetible): Define que cualquier tupla leída durante el transcurso de una transacción es bloqueada. De esta forma se soluciona, además de las lecturas sucias, el problema de las lecturas no repetibles. Aunque en dicho nivel se siguen dando las lecturas fantasmas.

Serializable (Lecturas en serie): Soluciona todos los problemas descritos. Para ello ordena las transacciones con el objetivo de que no entren en conflicto. Este nivel de aislamiento es bastante problemático ya que es, con diferencia, el que más sacrifica en rendimiento y concurrencia.

Niveles de aislamiento

Nivel de aislamiento	Lecturas fantasma	Lecturas no repetibles	Lecturas sucias
Serializable	No	No	No
Repeatable read	Si	No	No
Read committed	Si	Si	No
Read uncommitted	Si	Si	Si

Niveles de aislamiento

```
mysql> SELECT @@tx_isolation;
+-----------------+
| @@tx_isolation  |
+-----------------+
| REPEATABLE-READ |
+-----------------+
```

```
mysql> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

mysql> SELECT @@tx_isolation;
+-----------------+
| @@tx_isolation |
+-----------------+
| SERIALIZABLE   |
+-----------------+
```

Autocommit

MySQL también comete automáticamente las consultas que no son parte de una transacción. Los resultados de cualquier consulta ya sea ACTUALIZAR o INSERTAR que no esté precedida por un **START TRANSACTION** serán visibles inmediatamente para todas las conexiones.

```
mysql> SELECT @@autocommit;
+--------------+
| @@autocommit |
+--------------+
|            1 |
+--------------+
```

```
mysql> SET autocommit=0;

mysql> SELECT @@autocommit;
+--------------+
| @@autocommit |
+--------------+
|            0 |
+--------------+
```

Transacciones

Una transacción es un grupo secuencial de sentencias SQL.
Por ejemplo operaciones de actualización.

Una transacción sólo se confirma si cada operación individual dentro del grupo de sentencias tiene éxito.

Si al menos una de las operaciones falla, la transacción NO se completa.

ACID: Propiedades de las transacciones

- **Atomicidad:** asegura que todas las operaciones dentro de la unidad de trabajo se completen con éxito; de lo contrario, la transacción se cancela en el punto de falla y las operaciones anteriores se devuelven a su estado anterior.
- **Coherencia:** garantiza que la base de datos cambie correctamente los estados en una transacción confirmada con éxito.
- **Aislamiento:** permite que las transacciones operen independientemente y transparentes entre sí.
- **Durabilidad:** asegura que el resultado o efecto de una transacción confirmada persista en caso de una falla del sistema.

Transacciones

Inicio de transacción:

- START TRANSACTION // Inicio
- BEGIN WORK

Fin transacción:

- COMMIT // Confirma
- ROLLBACK . // Vuelve atrás

Las sentencias SQL entre las instrucciones de inicio y finalización forman parte de la transacción.

Ejemplo implementación de transacciones

```
START TRANSACTION;

    UPDATE accTable SET
    ledgerAmt=ledgerAmt-@transAmt WHERE
    customerId=1;

    UPDATE accTable SET
    ledgerAmt=ledgerAmt+@transAmt WHERE
    customerId=2;

COMMIT;
```

Transacciones

Procedimientos generales involucrados en la transacción.

- Comenzar la transacción con el comando SQL BEGIN WORK o START TRANSACTION .
- Ejecutar todas las sentencias SQL.
- Comprobar si todas las sentencias se ejecutan correctamente.
- Si la ejecución es válida, confirma mediante COMMIT; de lo contrario, revertir con el comando ROLLBACK.

Transacciones

La variable AUTOCOMMIT se establece como *verdadera* por defecto.

– Cambio a FALSE

```
SET AUTOCOMMIT=false;
SET AUTOCOMMIT=0;
```

--->Cambio a TRUE

```
SET AUTOCOMMIT=true;
SET AUTOCOMMIT=1;
```

Estado de AUTOCOMMIT

```
SELECT @@autocommit;
```

Transacciones con Control de Error

```
DELIMITER //
create procedure pro1(_clave integer, _dato integer)
BEGIN
 -- Declare acción por error
 DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
        rollback;
        SELECT 'Error en transaction' AS message;
 END;
        START TRANSACTION;
        update a set z = _dato where c = 1;
        insert into a(c,z) values (_clave,_dato);
        commit;
 end //
DELIMITER ;
```