

Transacciones

Índice

1.	Transacciones	3
2.	Uso de transacciones en MySQL.....	4

1. Transaccións

Unha transacción é un conxunto de instrucións SQL que se executan de maneira atómica ou indivisible como unha unidade, é dicir, ou se executan todas as instrucións ou non se executa ningunha. Se unha transacción ten éxito, todas as modificacións dos datos realizados durante a transacción se gardan na base de datos. Se unha transacción contén erros os cambios non se gardarán na base de datos.

Unha transacción debe cumprir as catro propiedades ACID:

- **Atomicidade:** asegura que se realizan todas as operacións ou ningunha, non pode quedar a medias.
- **Consistencia** ou integridade: asegura que só se empeza o que se pode acabar.
- **Illamento:** asegura que ningunha operación afecta a outras, con isto se asegura que varias transaccións sobre a mesma información sexan independentes e non xeren ningún tipo de erro.
- **Durabilidade:** asegura que unha vez realizada a operación, esta non poderá cambiar e permanecerán os cambios. Isto é, se o disco duro falla, o sistema aínda será capaz de lembrar todas as transaccións que se realizaron no sistema.

Exemplo: Unha transferencia económica na que debe substraerse unha cantidade dunha conta, facer unha serie de cálculos relacionados con comisións, intereses, etc. Todo iso debe ocorrer de maneira simultánea coma se fose unha soa operación, xa que doutro xeito xeráranse inconsistencias.

MySQL soporta distintos tipos de táboas tales como ISAM, MyISAM, InnoDB e BDB (Berkeley Database). **As táboas que permiten transaccións son do tipo InnoDB.** Están estruturadas de xeito distinto que MyISAM, xa que se almacenan nun só arquivo en lugar de tres e, ademais de transaccións, permiten definir regras de integridade referencial.

As transacción aportan seguridade ás bases de datos e a os procesos que se executan nelas. Ademais podemos dicir que as táboas que soportan transaccións son mais seguras.

No seguinte exemplo, se unha cantidade de diñeiro é transferida da conta dun cliente, requíranse polo menos dúas instrucións de actualización:

```
UPDATE tbl_contas SET balance = saldo - cantidadeTransferida WHERE idCliente='cc1';  
UPDATE tbl_contas SET balance = saldo + cantidadeTransferida WHERE idCliente = 'cc2',
```

Estas dúas consultas deben traballar ben, pero que sucede si ocorre algún imprevisto e cáese o sistema despois de que se executa a primeira instrución e a segunda aínda non se completou? O cliente 1 terá unha cantidade de diñeiro descontada da súa conta, e crerá que realizou o seu pago, con todo, o cliente 2 pensará que non se lle depositou o diñeiro que se lle debe. Neste sinxelo exemplo, móstrase a necesidade de que as consultas, ou ben sexan executadas de maneira conxunta ou que non se execute ningunha delas. É neste tipo de situacións onde as transaccións desempeñan un papel crucial.

Unha transacción ten dous finais posibles, **COMMIT** (execútanse todas as instrucións e gardamos os datos) e **ROLLBACK** (se produce un erro e non se gardan os cambios). Por defecto, MySQL trae activado o modo “*autocommit*”, polo que cando se realiza unha transacción (INSERT, UPDATE o DELETE) esta se confirma automaticamente. Para desactivar esta opción se debe executar o seguinte comando (**non recomendado**):

```
> SET AUTOCOMMIT=0;
/* ou tamén se pode desactivar para unha serie de comandos utilizando START
TRANSACTION. (Isto é o aconsellable). */
> START TRANSACTION;
> .....
> COMMIT;
```

2. Uso de transaccións en MySQL

Os pasos para usar transaccións en MySQL son:

- Iniciar unha transacción co uso da sentenza START TRANSACTION ou BEGIN.
- Actualizar, inserir ou eliminar rexistros na base de datos.
- No caso de querer reflectir os cambios na base de datos, completárase a transacción co uso da sentenza COMMIT. Unicamente cando se procesa un COMMIT os cambios feitos polas consultas serán permanentes.
- Se acontece algún problema, poderase facer ROLLBACK para cancelar os cambios que foron realizados polas consultas executadas ata o momento.

En táboas InnoDB, toda actividade do usuario prodúcese dentro dunha transacción. Se o modo de execución automática (autocommit) está activado, cada sentenza SQL conforma unha transacción individual por si mesma. MySQL sempre comeza unha nova conexión coa execución automática habilitada.

Se o modo de execución automática se inhabilitou con SET AUTOCOMMIT=0, entón pode considerarse que un usuario sempre ten unha transacción aberta. A transacción vixente remataríase cunha das sentenzas seguintes:

- Unha sentenza COMMIT, significa que os cambios feitos na transacción actual convértese en permanentes e vólvense visibles para os outros usuarios.
- Unha sentenza ROLLBACK, cancela todas as modificacións producidas na transacción actual.

Ambas sentenzas liberan todos os bloqueos *InnoDB* que se estableceron durante a transacción vixente.

Se a conexión ten a execución automática habilitada, o usuario pode igualmente levar a cabo unha transacción con varias sentenzas se a comeza explicitamente con START TRANSACTION ou BEGIN e a remata con COMMIT ou ROLLBACK.

O primeiro que se debe facer á hora de traballar con transaccións é comprobar o estado da variable “autocommit”:

```
SHOW VARIABLES LIKE 'autocommit';
```

Se ten o valor 1 podemos cambiala con SET. Outra opción é realizar os exemplos con START TRANSACTION ou BEGIN (isto é mais aconsellable).

No seguinte exemplo, creamos unha táboa, na base de datos “*bd_ProbaTransaccions*” do tipo InnoDB e inserimos algúns datos. Para crear unha táboa *InnoDB*, procederase co código SQL estándar CREATE TABLE, pero debemos especificar que se trata dunha táboa do tipo *InnoDB*. A táboa chamarase “*tbl_ProbaTransaccions*”. Primeiro activamos a base de datos “*bd_ProbaTransaccions*” coa instrución USE para despois crear a táboa e introducir algúns valores:

```
CREATE DATABASE `bd_ProbaTransaccions`;  
USE `bd_ProbaTransaccions`;  
SHOW VARIABLES LIKE 'AUTOCOMMIT';
```

	Variable_name	Value
►	autocommit	ON

```
USE `bd_ProbaTransaccions`;  
CREATE TABLE tbl_ProbaTransaccions (id INT NOT NULL PRIMARY KEY, s VARCHAR (30), si  
SMALLINT) ENGINE = InnoDB ;  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (1, "primeiro", NULL);  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (2, "segundo", NULL);  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (3, "terceiro", NULL);  
SELECT * FROM tbl_ProbaTransaccions;
```

	id	s	si
►	1	primeiro	NULL
	2	segundo	NULL
	3	terceiro	NULL

Unha vez cargada a táboa iniciamos unha transacción:

```
BEGIN;  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (4, "cuarto", NULL);  
ROLLBACK;  
SELECT * FROM tbl_ProbaTransaccions;
```

Ao executar un ROLLBACK, a transacción non será completada e os cambios realizados sobre a táboa non terán efecto:

	id	s	si
►	1	primeiro	NULL
	2	segundo	NULL
	3	terceiro	NULL

Se agora facemos un `SELECT` para mostrar os datos de `tbl_ProbaTransaccions`", comprobábase que non se produciu ningunha inserción.

Agora imos ver que sucede se perdemos a conexión ao servidor antes de que a transacción sexa completada.

```
BEGIN;  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (4, "cuarto", NULL);  
SELECT * FROM tbl_ProbaTransaccions;
```

	id	s	si
▶	1	primeiro	NULL
	2	segundo	NULL
	3	terceiro	NULL
	4	cuarto	NULL

--CAIDA DO SISTEMA SIMULADA: pechamos a conexión

Cando obtemos de novo a conexión, podemos verificar que o rexistro non se inseriu, xa que a transacción non foi completada.

```
USE `bd_ProbaTransaccions`;  
SELECT * FROM bl_ProbaTransaccions;
```

	id	s	si
▶	1	primeiro	NULL
	2	segundo	NULL
	3	terceiro	NULL

Unha vez que facemos un `COMMIT`, a transacción é completada e todas as sentenzas SQL que foron executadas previamente afectan de maneira permanente ás táboas da base de datos.

Hai instrucións SQL que dadas as súas características, implican confirmación automática. Estas son:

- Instrucións do DDL que definen ou modifican obxectos da base de datos (`CREATE`, `ALTER`, `DROP`, `RENAME` e `TRUNCATE`).
- Instrucións que modifican táboas da base de datos administrativa chamada *mysql*, (`GRANT` e `REVOKE`).
- Instrucións de control de transaccións e bloqueo de táboas (`START TRANSACTION`, `BEGIN`, `LOCK` e `UNLOCK`).
- Instrucións de carga de datos (`LOAD DATA`), para importar datos desde arquivo.
- Instrucións de administración de táboas (`ANALIZE`, `CHECK`, `OPTIMIZE` e `REPAIR`).

Existen ademais dous comandos para a xestión de transaccións:

- **SAVEPOINT** id: é unha instrución que permite nomear certo paso nun conxunto de instrucións dunha transacción.
- **ROLLBACK TO SAVEPOINT** id: permite desfacer o conxunto de operacións realizadas a partir do identificador de *savepoint*.
- **RELEASE SAVEPOINT** id: elimina ou libera o *savepoint* creado.

Toda operación **COMMIT** ou **ROLLBACK** sen argumentos eliminará todos os *savepoints* creados.

```
START TRANSACTION;  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (6, "sexto", NULL);  
SAVEPOINT un;  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (7, "setimo", NULL);  
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (8, "oitavo", NULL);  
ROLLBACK TO un;  
SELECT * FROM tbl_ProbaTransaccions;
```

Este sinxelo exemplo fai que só se insiran realmente (se confirmen) os datos do primeiro **INSERT**. Comprobámolo cunha sentenza **SELECT**.

id	s	si
1	primeiro	NULL
2	segundo	NULL
3	terceiro	NULL
4	cuarto	NULL
5	quinto	NULL
6	sexto	NULL