2. Transacciones

2.1. Transacciones

Una transacción es un conjunto de instrucciones SQL que se ejecutan de manera atómica o indivisible como una unidad. Es decir, o se ejecutan todas las instrucciones o no se ejecuta ninguna. Si una transacción tiene éxito, todas las modificaciones de los datos realizados durante la transacción se guardan en la base de datos. Si una transacción contiene errores, los cambios no se guardarán en la base de datos.

Una transacción debe cumplir las cuatro propiedades ACID (*Atomicity, Consistency, Isolation, Durability*):

- **Atomicidad**: asegura que se realizan todas las operaciones o ninguna. No puede quedar a medias.
- **Consistencia** o integridad: asegura que sólo se empieza lo que se puede acabar.
- **Aislamiento**: asegura que ninguna operación afecta a otras. Con esto se asegura que varias transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- **Durabilidad**: asegura que una vez realizada la operación, esta no podrá cambiar y permanecerán los cambios.

Un ejemplo típico de transacción es una transferencia económica en la que se debe sustraer una cantidad de una cuenta bancaria, hacer una serie de cálculos relacionados con comisiones, intereses, etc. Todo esto debe ocurrir de forma simultánea como si fuese una sola operación, ya que de otra forma se generarían inconsistencias.

Una de las características de los sistemas gestores es que se permite o no el uso de transacciones en sus tablas. En el caso de MySQL, esto depende del tipo de tabla o motor utilizado. MySQL soporta distintos tipos de tablas, tales como ISAM, MyISAM, InnoDB y BDB (*Berkeley Database*). Las tablas que permiten transacciones son del tipo InnoDB.

Las transacciones permiten que las bases de datos sean más fiables. Si disponemos de una serie de operaciones SQL que se deben ejecutar en conjunto, con el uso de transacciones podemos tener la certeza de que nunca nos quedaremos a medio camino de su ejecución.

Las tablas que soportan transacciones son mucho más seguras y fáciles de recuperar en el caso de producirse algún fallo en el servidor, ya que las instrucciones se ejecutan o no en su totalidad. Por otro lado, las transacciones pueden aumentar el tiempo de proceso de instrucciones.

En el siguiente ejemplo, si una cantidad de dinero es transferida desde la cuenta bancaria de un cliente, se requieren por lo menos dos instrucciones de actualización:

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

¿Qué ocurre si se produce algún imprevisto y se cae el sistema después de que se ejecute la primera instrucción y la segunda aún no se haya completado? El cliente 1 tendrá menos dinero en su cuenta

y creerá que se realizó el pago. Por otra parte, la cantidad transferida no se habrá abonado en la cuenta del cliente 2, así que pensará que no se le depositó el dinero que se le debe. Este es un ejemplo claro de la necesidad de que las sentencias se ejecuten o bien de forma conjunta o bien que no se ejecute ninguna. Es en este tipo de situaciones donde las transacciones desempeñan un papel crucial.

Una transacción tiene dos finales posibles, COMMIT (se ejecutan todas las instrucciones de forma conjunta y guardamos los datos) y ROLLBACK (se produce un error y no se guardan los cambios). Por defecto, MySQL trae activado el modo *autocommit*, por lo que cuando se realiza una transacción (INSERT, UPDATE o DELETE) esta se confirma automáticamente. Para desactivar esta opción se debe ejecutar el siguiente comando (no recomendado):

```
> SET AUTOCOMMIT=0;
```

Aunque lo recomendable es utilizar START TRANSACTION:

```
> START TRANSACTION;
> .....
> COMMIT;
```

Uso de transacciones en MySQL

Los pasos para usar transacciones en MySQL son:

- Iniciar una transacción con el uso de la sentencia START TRANSACTION o BEGIN.
- Actualizar, insertar o eliminar registros en la base de datos.
- En el caso de querer reflejar los cambios en la base de datos, se completará la transacción con el uso de la sentencia COMMIT. Únicamente los cambios serán permanentes cuando se procesa un COMMIT.
- Si ocurre algún problema, se puede hacer ROLLBACK para cancelar los cambios que fueron realizados por las consultas ejecutadas hasta el momento.

En tabla InnoDB, toda la actividad del usuario se produce dentro de una transacción. Si el modo de ejecución automática (*autocommit*) está activado, cada sentencia SQL conforma una transacción individual por sí misma. MySQL siempre comienza una nueva conexión con la ejecución automática habilitada.

Si el modo de ejecución automática se inhabilita con SET AUTOCOMMIT=0, entonces se puede considerar que un usuario siempre tiene una transacción abierta. La transacción vigente terminaría con una de las siguientes sentencias:

- Una sentencia COMMIT, que significa que los cambios hechos en la transacción actual se convierten en permanentes y se vuelven visibles para otros usuarios.
- Una sentencia ROLLBACK, que cancela todas las modificaciones producidas en la transacción actual.

Si la conexión tiene la ejecución automática habilitada, el usuario puede igualmente llevar a cabo una transacción con varias sentencias si la empieza explícitamente con START TRANSACTION o BEGIN y la termina con COMMIT o ROLLBACK.

Lo primero que se debe hacer a la hora de trabajar con transacciones es comprobar el estado de la variable *autocommit*:

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

Si tiene el valor 'ON' la podemos desactivar con SET, aunque la mejor opción es realizar los ejemplos con START TRANSACTION.

En el siguiente ejemplo, creamos una tabla en la base de datos *bd_ProbaTransaccions* e insertamos algunos datos:

```
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

Una vez cargados los datos de la tabla, iniciamos una transacción:

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

Al ejecutar un ROLLBACK, la transacción no será completada y los cambios realizados sobre la tabla no tendrán efecto:

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

En el siguiente ejemplo, se repite la sentencia INSERT ejecutada anteriormente y se añade otra, pero haciendo un COMMIT.

```
START TRANSACTION;
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (4, "cuarto", NULL);
COMMIT;
INSERT INTO tbl_ProbaTransaccions ( id, s, si) VALUES (5, "quinto", NULL);
COMMIT;
SELECT * FROM  tbl_ProbaTransaccions;
```

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

Una vez que hacemos un COMMIT, la transacción se completa y todas las sentencias SQL que fueron ejecutadas previamente afectan de forma permanente a las tablas de las bases de datos.