Boletín de ejemplos de transacciones

Aunque la variable AUTOCOMMIT está activada por defecto al inicio de una sesión SQL, podemos configurarlo para indicar si queremos trabajar con transacciones implícitas o explíticas.

Podemos consultar el valor actual de AUTOCOMMIT haciendo:

SELECT @@AUTOCOMMIT. También podemos hacerlo con: SHOW VARIABLES LIKE 'autocommit';

Para desactivar la variable AUTOCOMMIT hacemos:

SET AUTOCOMMIT = 0;

Si hacemos esto siempre tendríamos una transacción abierta y los cambios sólo se aplicarían en la base de datos ejecutando la sentencia COMMIT de forma explícita.

Para activar la variable AUTOCOMMIT hacemos:

SET AUTOCOMMIT = 1;

Para poder trabajar con transacciones en MySQL es necesario utilizar InnoDB

Ejemplo 1:

```
DROP DATABASE IF EXISTS test;
CREATE DATABASE test CHARACTER SET utf8mb4;
USE test;

CREATE TABLE cliente (
   id INT UNSIGNED PRIMARY KEY,
   nombre CHAR (20)
);

START TRANSACTION;
INSERT INTO cliente VALUES (1, 'Pepe');
COMMIT;

-- 1. ¿Qué devolverá esta consulta?
SELECT *
FROM cliente;

SET AUTOCOMMIT=0;
INSERT INTO cliente VALUES (2, 'Maria');
INSERT INTO cliente VALUES (20, 'Juan');
DELETE FROM cliente WHERE nombre = 'Pepe';

-- 2. ¿Qué devolverá esta consulta?
SELECT *
FROM cliente;

ROLLBACK;

-- 3. ¿Qué devolverá esta consulta?
SELECT *
FROM cliente;
```

Ejemplo 2:

```
DROP DATABASE IF EXISTS test;
CREATE DATABASE test CHARACTER SET utf8mb4;
USE test;

CREATE TABLE cuentas (
   id INTEGER UNSIGNED PRIMARY KEY,
   saldo DECIMAL(11,2) CHECK (saldo >= 0)
);

INSERT INTO cuentas VALUES (1, 1000);
INSERT INTO cuentas VALUES (2, 2000);
INSERT INTO cuentas VALUES (3, 0);

-- 1. Consultamos el estado actual de las cuentas
SELECT *
FROM cuentas;

-- 2. Suponga que queremos realizar una transferencia de dinero entre dos cuentas bancarias con la siguiente transacción:
START TRANSACTION;
UPDATE cuentas SET saldo = saldo - 100 WHERE id = 1;
UPDATE cuentas SET saldo = saldo + 100 WHERE id = 2;
COMMIT;

-- 3. ¿Qué devolverá esta consulta?
SELECT *
FROM cuentas;

-- 4. Suponga que queremos realizar una transferencia de dinero entre dos cuentas bancarias con la siguiente transacción y una de las dos cuentas no existe:
START TRANSACTION;
UPDATE cuentas SET saldo = saldo - 100 WHERE id = 9999;
UPDATE cuentas SET saldo = saldo + 100 WHERE id = 2;
COMMIT;

-- 5. ¿Qué devolverá esta consulta?
SELECT *
FROM cuentas;

-- 6. Suponga que queremos realizar una transferencia de dinero entre dos cuentas bancarias con la siguiente transacción y la cuenta origen no tiene saldo:
START TRANSACTION;
UPDATE cuentas SET saldo = saldo - 100 WHERE id = 3;
UPDATE cuentas SET saldo = saldo + 100 WHERE id = 2;
COMMIT;

-- 7. ¿Qué devolverá esta consulta?
SELECT *
FROM cuentas;
```

Ejemplo 3:

```sql
DROP DATABASE IF EXISTS test;
CREATE DATABASE test CHARACTER SET utf8mb4;
USE test;

CREATE TABLE producto (
  id INT UNSIGNED PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  precio DOUBLE
);

INSERT INTO producto (id, nombre) VALUES (1, 'Primero');
INSERT INTO producto (id, nombre) VALUES (2, 'Segundo');
INSERT INTO producto (id, nombre) VALUES (3, 'Tercero');

-- 1. Comprobamos las filas que existen en la tabla
SELECT *
FROM producto;

-- 2. Ejecutamos una transacción que incluye un SAVEPOINT
START TRANSACTION;
INSERT INTO producto (id, nombre) VALUES (4, 'Cuarto');
SAVEPOINT sp1;
INSERT INTO producto (id, nombre) VALUES (5, 'Cinco');
INSERT INTO producto (id, nombre) VALUES (6, 'Seis');
ROLLBACK TO sp1;

-- 3. ¿Qué devolverá esta consulta?
SELECT *
FROM producto;
```