12 SQL - TCL

Transaction Control Language (TCL) in SQL

- TCL commands are used to manage transactions in a database
- A transaction is a sequence of operations performed as a single logical unit of work.
- If any operation within a transaction fails, the entire transaction can be rolled back to ensure data integrity

The primary TCL commands are:

- **COMMIT**: Saves the changes made by the transaction permanently to the database
- **ROLLBACK**: Reverts the changes made by the transaction to the last committed state
- **SAVEPOINT**: Sets a point within a transaction to which a rollback can be performed

Example Scenario: Employee Database

Let's consider a simple `Employee` database with a table `employees`:

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10, 2)
);
```

```
mysql> CREATE DATABASE DEMO;
Query OK, 1 row affected (0.31 sec)

mysql> USE DEMO;
Database changed
mysql> CREATE TABLE employees (
    ->     emp_id INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     department VARCHAR(50),
    ->     salary DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (2.34 sec)

mysql> DESC employees;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| emp_id     | int           | NO   | PRI | NULL    |       |
| name       | varchar(50)   | YES  |     | NULL    |       |
| department | varchar(50)   | YES  |     | NULL    |       |
| salary     | decimal(10,2) | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Step 1: Inserting Records and Using Savepoints

```
-- Start a transaction
START TRANSACTION;
```

```
INSERT INTO employees (emp_id, name, department, salary)
VALUES (1, 'John Doe', 'IT', 60000);

-- Set SAVEPOINT A
SAVEPOINT A;
```

```
INSERT INTO employees (emp_id, name, department, salary)
VALUES (2, 'Jane Smith', 'HR', 55000);

-- Set SAVEPOINT B
SAVEPOINT B;
```

```
INSERT INTO employees (emp_id, name, department, salary)
VALUES (3, 'Alice Johnson', 'Finance', 62000);

-- Set SAVEPOINT C
SAVEPOINT C;
```

```sql
INSERT INTO employees (emp_id, name, department, salary)
VALUES (4, 'Bob Brown', 'IT', 58000);
```

```
mysql> SELECT*FROM employees;
+--------+---------------+------------+----------+
| emp_id | name          | department | salary   |
+--------+---------------+------------+----------+
|      1 | John Doe      | IT         | 60000.00 |
|      2 | Jane Smith    | HR         | 55000.00 |
|      3 | Alice Johnson | Finance    | 62000.00 |
|      4 | Bob Brown     | IT         | 58000.00 |
+--------+---------------+------------+----------+
4 rows in set (0.00 sec)
```

Step 2: Using ROLLBACK

- Let's say we made a mistake with the last insertion and want to undo it:

```sql
ROLLBACK TO SAVEPOINT C;
```

```
mysql> ROLLBACK TO SAVEPOINT C;
Query OK, 0 rows affected (0.20 sec)

mysql> SELECT*FROM employees;
+--------+---------------+------------+----------+
| emp_id | name          | department | salary   |
+--------+---------------+------------+----------+
|      1 | John Doe      | IT         | 60000.00 |
|      2 | Jane Smith    | HR         | 55000.00 |
|      3 | Alice Johnson | Finance    | 62000.00 |
+--------+---------------+------------+----------+
3 rows in set (0.00 sec)
```

```sql
-- Rollback to SAVEPOINT B
ROLLBACK TO SAVEPOINT B;
```

```
mysql> SELECT*FROM employees;
+--------+------------+------------+----------+
| emp_id | name       | department | salary   |
+--------+------------+------------+----------+
|      1 | John Doe   | IT         | 60000.00 |
|      2 | Jane Smith | HR         | 55000.00 |
+--------+------------+------------+----------+
2 rows in set (0.00 sec)
```

- If you want to undo all changes since the beginning of the transaction:

```sql
-- Rollback to the start of the transaction
```

```
ROLLBACK;
```

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT*FROM employees;
Empty set (0.00 sec)
```

Step 3: COMMIT to Save Changes

- If everything looks good, you can save all the changes:

```
COMMIT;
```

- This will save all the changes made after the last commit (or start of the transaction) permanently in the database.

```
mysql> SELECT*FROM employees;
+--------+--------------+------------+----------+
| emp_id | name         | department | salary   |
+--------+--------------+------------+----------+
|      1 | John Doe     | IT         | 60000.00 |
|      2 | Jane Smith   | HR         | 55000.00 |
|      3 | Alice Johnson | Finance    | 62000.00 |
|      4 | Bob Brown    | IT         | 58000.00 |
+--------+--------------+------------+----------+
4 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT C;
ERROR 1305 (42000): SAVEPOINT C does not exist
mysql> SELECT*FROM employees;
+--------+--------------+------------+----------+
| emp_id | name         | department | salary   |
+--------+--------------+------------+----------+
|      1 | John Doe     | IT         | 60000.00 |
|      2 | Jane Smith   | HR         | 55000.00 |
|      3 | Alice Johnson | Finance    | 62000.00 |
|      4 | Bob Brown    | IT         | 58000.00 |
+--------+--------------+------------+----------+
4 rows in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT*FROM employees;
+--------+--------------+------------+----------+
| emp_id | name         | department | salary   |
+--------+--------------+------------+----------+
|      1 | John Doe     | IT         | 60000.00 |
|      2 | Jane Smith   | HR         | 55000.00 |
|      3 | Alice Johnson | Finance    | 62000.00 |
|      4 | Bob Brown    | IT         | 58000.00 |
+--------+--------------+------------+----------+
4 rows in set (0.00 sec)
```

Autocommit in SQL

- **Autocommit** is a mode in SQL where each individual SQL statement is treated as a transaction and is automatically committed after it is executed
- In most database systems, autocommit is turned on by default

Example of Autocommit:

- Autocommit is usually ON by default

```
INSERT INTO employees (emp_id, name, department, salary)
VALUES (5, 'David Green', 'Marketing', 50000);
```

- This insertion is automatically committed to the database
- If autocommit is disabled, you would need to manually commit each transaction:

```
-- Disable autocommit
SET AUTOCOMMIT = 0;
```

```
-- Start a transaction
START TRANSACTION;
```

```
INSERT INTO employees (emp_id, name, department, salary)
VALUES (6, 'Emily White', 'Sales', 48000);
```

```
-- Changes are not saved until you commit
COMMIT;
```

```
-- Re-enable autocommit
SET AUTOCOMMIT = 1;
```