MYSQL

(11-03-2024)

1. **DQL** *(Data Query Language)* : Used to retrieve data from databases. (SELECT)
2. **DDL** *(Data Definition Language)* : Used to create, alter, and delete database objects like tables, indexes, etc. (CREATE, DROP, ALTER, RENAME, TRUNCATE)
3. **DML** *(Data Manipulation Language)*: Used to modify the database. (INSERT, UPDATE, DELETE)
4. **DCL** *(Data Control Language)*: Used to grant & revoke permissions. (GRANT, REVOKE)
5. **TCL** *(Transaction Control Language)*: Used to manage transactions. (COMMIT, ROLLBACK, START TRANSACTIONS, SAVEPOINT)

---

1. Data Definition Language (DDL)

Data Definition Language (DDL) is a subset of SQL (Structured Query Language) responsible for defining and managing the structure of databases and their objects.

DDL commands enable you to create, modify, and delete database objects like tables, indexes, constraints, and more.

Key DDL Commands are:

- **CREATE TABLE:**
  - Used to create a new table in the database.
  - Specifies the table name, column names, data types, constraints, and more.
  - Example:
    CREATE TABLE employees (id INT PRIMARY KEY, name VARCHAR(50), salary DECIMAL(10, 2));

- **ALTER TABLE:**
  - Used to modify the structure of an existing table.
  - You can add, modify, or drop columns, constraints, and more.
  - Example: ALTER TABLE employees ADD COLUMN email VARCHAR(100);

- **DROP TABLE:**
  - Used to delete an existing table along with its data and structure.
  - Example: DROP TABLE employees;

- **CREATE INDEX:**
    - Used to create an index on one or more columns in a table.
    - Improves query performance by enabling faster data retrieval.
    - Example: CREATE INDEX idx_employee_name ON employees (name);
- **DROP INDEX:**
    - Used to remove an existing index from a table.
    - Example: DROP INDEX idx_employee_name;
- **CREATE CONSTRAINT:**
    - Used to define constraints that ensure data integrity.
    - Constraints include PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, and CHECK.
    - Example: ALTER TABLE orders ADD CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES customers(id);
- **DROP CONSTRAINT:**
    - Used to remove an existing constraint from a table.
    - Example: ALTER TABLE orders DROP CONSTRAINT fk_customer;
- **TRUNCATE TABLE:**
    - Used to delete the data inside a table, but not the table itself.
    - Syntax – TRUNCATE TABLE table_name

2. DATA QUERY/RETRIEVAL LANGUAGE (DQL or DRL)

DQL (Data Query Language) is a subset of SQL focused on retrieving data from databases.

The SELECT statement is the foundation of DQL and allows us to extract specific columns from a table.

- **SELECT:**

The SELECT statement is used to select data from a database.

Syntax: SELECT column1, column2, ... FROM table_name;

Here, column1, column2, ... are the field names of the table.

If you want to select all the fields available in the table, use the following syntax:
SELECT * FROM table_name;

Ex: SELECT CustomerName, City FROM Customers;

- **WHERE:**

The WHERE clause is used to filter records.

Syntax: SELECT column1, column2, ... FROM table_name WHERE condition;

Ex: SELECT * FROM Customers WHERE Country='Mexico';

Operators used in WHERE are:

= : Equal
> : Greater than
< : Less than
>= : Greater than or equal
<= : Less than or equal
<> : Not equal.

Note: In some versions of SQL this operator may be written as !=

- **AND, OR and NOT:**

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) is NOT TRUE.

Syntax:

SELECT column1, column2, ... FROM table_name WHERE condition1 AND condition2 AND condition3 ...;

SELECT column1, column2, ... FROM table_name WHERE condition1 OR condition2 OR condition3 ...;

SELECT column1, column2, ... FROM table_name WHERE NOT condition;

Example:

SELECT * FROM Customers WHERE Country='India' AND City='Japan';

SELECT * FROM Customers WHERE Country='America' AND (City='India' OR City='Korea');

- **DISTINCT:**

Removes duplicate rows from query results.

Syntax: SELECT DISTINCT column1, column2 FROM table_name;

- **LIKE**:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

Example: SELECT * FROM employees WHERE first_name LIKE 'J%';

WHERE CustomerName LIKE 'a%'
- Finds any values that start with "a"

WHERE CustomerName LIKE '%a'
- Finds any values that end with "a"

WHERE CustomerName LIKE '%or%'
- Finds any values that have "or" in any position

WHERE CustomerName LIKE '_r%'
- Finds any values that have "r" in the second position

WHERE CustomerName LIKE 'a_%'

- Finds any values that start with "a" and are at least 2 characters in length

WHERE CustomerName LIKE 'a__%'

- Finds any values that start with "a" and are at least 3 characters in length

WHERE ContactName LIKE 'a%o'

- Finds any values that start with "a" and ends with "o"

- **IN:**

Filters results based on a list of values in the WHERE clause.

Example: SELECT * FROM products WHERE category_id IN (1, 2, 3);

- **BETWEEN**:

Filters results within a specified range in the WHERE clause.

Example: SELECT * FROM orders WHERE order_date BETWEEN '2023-01-01' AND '2023-06-30';

- **IS NULL**:

Checks for NULL values in the WHERE clause.

Example: SELECT * FROM customers WHERE email IS NULL;

- **AS**:

Renames columns or expressions in query results.

Example: SELECT first_name AS "First Name", last_name AS "Last Name" FROM employees;

- **ORDER BY**

The ORDER BY clause allows you to sort the result set of a query based on one or more columns.

Basic Syntax:

- The ORDER BY clause is used after the SELECT statement to sort query results.

- Syntax: SELECT column1, column2 FROM table_name ORDER BY column1 [ASC|DESC];

Ascending and Descending Order:

- By default, the ORDER BY clause sorts in ascending order (smallest to largest).
- You can explicitly specify descending order using the DESC keyword.
- Example: SELECT product_name, price FROM products ORDER BY price DESC;

Sorting by Multiple Columns:

- You can sort by multiple columns by listing them sequentially in the ORDER BY clause.
- Rows are first sorted based on the first column, and for rows with equal values, subsequent columns are used for further sorting.
- Example: SELECT first_name, last_name FROM employees ORDER BY last_name, first_name;

Sorting by Expressions:

- It's possible to sort by calculated expressions, not just column values.
- Example: SELECT product_name, price, price * 1.1 AS discounted_price FROM products ORDER BY discounted_price;

Sorting NULL Values:

- By default, NULL values are considered the smallest in ascending order and the largest in descending order.
- You can control the sorting behaviour of NULL values using the NULLS FIRST or NULLS LAST options.
- Example: SELECT column_name FROM table_name ORDER BY column_name NULLS LAST;

Sorting by Position:

- Instead of specifying column names, you can sort by column positions in the ORDER BY clause.
- Example: SELECT product_name, price FROM products ORDER BY 2 DESC, 1 ASC;

- **GROUP BY**

The GROUP BY clause in SQL is used to group rows from a table based on one or more columns.

Syntax:

- The GROUP BY clause follows the SELECT statement and is used to group rows based on specified columns.

- Syntax: SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1;

- Aggregation Functions:
    - Aggregation functions (e.g., COUNT, SUM, AVG, MAX, MIN) are often used with GROUP BY to calculate values for each group.
    - Example: SELECT department, AVG(salary) FROM employees GROUP BY department;
- Grouping by Multiple Columns:
    - You can group by multiple columns by listing them in the GROUP BY clause.
    - This creates a hierarchical grouping based on the specified columns.
    - Example: SELECT department, gender, AVG(salary) FROM employees GROUP BY department, gender;

- HAVING Clause:
    - The HAVING clause is used with GROUP BY to filter groups based on aggregate function results.
    - It's similar to the WHERE clause but operates on grouped data.
    - Example: SELECT department, AVG(salary) FROM employees GROUP BY department HAVING AVG(salary) > 50000;

- Combining GROUP BY and ORDER BY:
    - You can use both GROUP BY and ORDER BY in the same query to control the order of grouped results.
    - Example: SELECT department, COUNT(*) FROM employees GROUP BY department ORDER BY COUNT(*) DESC;

- **AGGREGATE FUNCTIONS**

These are used to perform calculations on groups of rows or entire result sets. They provide insights into data by summarising and processing information.

Common Aggregate Functions:

- COUNT():
  Counts the number of rows in a group or result set.

- SUM():
  Calculates the sum of numeric values in a group or result set.

- AVG():

  Computes the average of numeric values in a group or result set.

- MAX():
  Finds the maximum value in a group or result set.

- MIN():
  Retrieves the minimum value in a group or result set.

3. DATA MANIPULATION LANGUAGE

Data Manipulation Language (DML) in SQL encompasses commands that manipulate data within a database. DML allows you to insert, update, and delete records, ensuring the accuracy and currency of your data.

- **INSERT:**

- The INSERT statement adds new records to a table.
- Syntax: INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);

- Example: INSERT INTO employees (first_name, last_name, salary) VALUES ('John', 'Doe', 50000);

- **UPDATE**:

- The UPDATE statement modifies existing records in a table.
- Syntax: UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;
- Example: UPDATE employees SET salary = 55000 WHERE first_name = 'John';

- **DELETE**:

- The DELETE statement removes records from a table.
- Syntax: DELETE FROM table_name WHERE condition;
- Example: DELETE FROM employees WHERE last_name = 'Doe';

---

4. Data Control Language (DCL)

Data Control Language focuses on the management of access rights, permissions, and security-related aspects of a database system.

DCL commands are used to control who can access the data, modify the data, or perform administrative tasks within a database.

DCL is an important aspect of database security, ensuring that data remains protected and only authorised users have the necessary privileges.

There are two main DCL commands in SQL: GRANT and REVOKE.

1. GRANT:

The GRANT command is used to provide specific privileges or permissions to users or roles. Privileges can include the ability to perform various actions on tables, views, procedures, and other database objects.

Syntax:

```
GRANT privilege_type
ON object_name
TO user_or_role;
```

In this syntax:

- privilege_type refers to the specific privilege or permission being granted (e.g., SELECT, INSERT, UPDATE, DELETE).
- object_name is the name of the database object (e.g., table, view) to which the privilege is being granted.
- user_or_role is the name of the user or role that is being granted the privilege.

Example: Granting SELECT privilege on a table named "Employees" to a user named "Analyst":

GRANT SELECT ON Employees TO Analyst;

2. REVOKE:

The REVOKE command is used to remove or revoke specific privileges or permissions that have been previously granted to users or roles.

Syntax:

REVOKE privilege_type
ON object_name

FROM user_or_role;

In this syntax:

- privilege_type is the privilege or permission being revoked.
- object_name is the name of the database object from which the privilege is being revoked.
- user_or_role is the name of the user or role from which the privilege is being revoked.

Example: Revoking the SELECT privilege on the "Employees" table from the "Analyst" user:

REVOKE SELECT ON Employees FROM Analyst;

DCL and Database Security:

DCL plays a crucial role in ensuring the security and integrity of a database system.

By controlling access and permissions, DCL helps prevent unauthorised users from tampering with or accessing sensitive data. Proper use of GRANT and REVOKE commands ensures that only users who require specific privileges can perform certain actions on database objects.

5. Transaction Control Language (TCL)

Transaction Control Language (TCL) deals with the management of transactions within a database.
TCL commands are used to control the initiation, execution, and termination of transactions, which are sequences of one or more SQL statements that are executed as a single unit of work.
Transactions ensure data consistency, integrity, and reliability in a database by grouping related operations together and either committing or rolling back changes based on the success or failure of those operations.

There are three main TCL commands in SQL: COMMIT, ROLLBACK, and SAVEPOINT.

1. COMMIT:

The COMMIT command is used to permanently save the changes made during a transaction.

It makes all the changes applied to the database since the last COMMIT or ROLLBACK command permanent.
Once a COMMIT is executed, the transaction is considered successful, and the changes are made permanent.

Example: Committing changes made during a transaction:

```
UPDATE Employees
SET Salary = Salary * 1.10
WHERE Department = 'Sales';

COMMIT;
```

2. ROLLBACK:

The ROLLBACK command is used to undo changes made during a transaction.
It reverts all the changes applied to the database since the transaction began.

ROLLBACK is typically used when an error occurs during the execution of a transaction, ensuring that the database remains in a consistent state.

Example: Rolling back changes due to an error during a transaction:

```
BEGIN;

UPDATE Inventory
SET Quantity = Quantity - 10
WHERE ProductID = 101;

-- An error occurs here

ROLLBACK;
```

3. SAVEPOINT:

The SAVEPOINT command creates a named point within a transaction, allowing you to set a point to which you can later ROLLBACK if needed.

SAVEPOINTs are useful when you want to undo part of a transaction while preserving other changes.

Syntax: SAVEPOINT savepoint_name;

Example: Using SAVEPOINT to create a point within a transaction:

BEGIN;

```
UPDATE Accounts
SET Balance = Balance - 100
WHERE AccountID = 123;

SAVEPOINT before_withdrawal;

UPDATE Accounts
SET Balance = Balance + 100
WHERE AccountID = 456;

-- An error occurs here

ROLLBACK TO before_withdrawal;

-- The first update is still applied

COMMIT;
```

TCL and Transaction Management:

Transaction Control Language (TCL) commands are vital for managing the integrity and consistency of a database's data.
They allow you to group related changes into transactions, and in the event of errors, either commit those changes or roll them back to maintain data integrity.
TCL commands are used in combination with Data Manipulation Language (DML) and other SQL commands to ensure that the database remains in a reliable state despite unforeseen errors or issues.