**SQL**

SQL, or Structured Query Language, is a standard programming language used to manage and manipulate relational databases. It allows users to interact with databases by performing tasks such as querying data, updating data, defining structures, and controlling access to data. Here are some key points about SQL:

1. \*\*Database Management\*\*: SQL is used to manage relational database management systems (RDBMS) such as MySQL, PostgreSQL, SQLite, SQL Server, and Oracle Database.

2. \*\*Data Manipulation\*\*: SQL provides commands for manipulating data stored in a database. These commands include SELECT (for querying data), INSERT (for adding new records), UPDATE (for modifying existing records), and DELETE (for removing records).

3. \*\*Data Definition\*\*: SQL allows users to define the structure of databases, including creating and altering tables, specifying constraints (such as primary keys and foreign keys), and defining indexes for performance optimization.

4. \*\*Data Control\*\*: SQL includes commands for controlling access to data within databases, including GRANT (for granting permissions) and REVOKE (for revoking permissions) statements.

5. \*\*Querying\*\*: SQL queries are used to retrieve specific data from a database based on specified criteria. Queries can be simple or complex and can involve filtering, sorting, aggregating, and joining data from multiple tables.

6. \*\*Transactions\*\*: SQL supports transactions, which are sequences of operations that are treated as a single unit of work. Transactions ensure the integrity of data by allowing operations to be either completed in full or rolled back if an error occurs.

7. \*\*Views and Stored Procedures\*\*: SQL allows users to create views, which are virtual tables derived from the underlying database tables, and stored procedures, which are precompiled sets of SQL statements that can be executed as a single unit.

8. \*\*Data Types\*\*: SQL supports various data types, including numeric, string, date/time, and binary types, which can be used to define the structure of database tables and the types of values stored in them.

Overall, SQL is a powerful and versatile language that is widely used for managing and querying relational databases in a wide range of applications, from simple data storage and retrieval to complex business intelligence and data analysis tasks.

**DML**

DML stands for Data Manipulation Language, and it comprises a set of SQL statements used to manipulate data stored in a database. The main DML statements in SQL are:

1. \*\*INSERT\*\*: Used to add new records (rows) into a table.

```sql

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

```

2. \*\*UPDATE\*\*: Used to modify existing records (rows) in a table.

```sql

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

```

3. \*\*DELETE\*\*: Used to remove records (rows) from a table.

```sql

DELETE FROM table\_name

WHERE condition;

```

4. \*\*MERGE\*\* (some databases): Combines INSERT, UPDATE, and DELETE operations into a single statement based on a condition.

```sql

MERGE INTO target\_table USING source\_table

ON (condition)

WHEN MATCHED THEN

UPDATE SET column1 = value1, column2 = value2

WHEN NOT MATCHED THEN

INSERT (column1, column2, ...)

VALUES (value1, value2, ...);

```

These statements allow you to perform various operations on the data stored in your database, such as adding new data, updating existing data, or removing unwanted data. They are fundamental for managing the content of your database tables.

TCL statements

TCL stands for Transaction Control Language, and it consists of SQL statements used to manage transactions within a database. The main TCL statements in SQL are:

1. \*\*COMMIT\*\*: Saves all changes made during the current transaction to the database. It makes the changes permanent and visible to other users.

```sql

COMMIT;

```

2. \*\*ROLLBACK\*\*: Undoes all changes made during the current transaction and restores the database to its state before the transaction began.

```sql

ROLLBACK;

```

3. \*\*SAVEPOINT\*\*: Sets a named point in the transaction to which you can later roll back if needed. It allows you to create intermediate points within a transaction.

```sql

SAVEPOINT savepoint\_name;

```

4. \*\*ROLLBACK TO SAVEPOINT\*\*: Rolls back the transaction to the specified savepoint, undoing any changes made after that point.

```sql

ROLLBACK TO SAVEPOINT savepoint\_name;

```

5. \*\*RELEASE SAVEPOINT\*\*: Removes a savepoint from the current transaction. It frees up resources and cannot be rolled back to.

```sql

RELEASE SAVEPOINT savepoint\_name;

```

These statements are crucial for managing the integrity and consistency of data within a database when dealing with transactions. They allow you to control when changes made by transactions are permanently saved or discarded.

DDL statement

DDL (Data Definition Language) statements in SQL are used to define, modify, and manage the structure of database objects such as tables, indexes, views, and schemas. These statements are responsible for creating, altering, and dropping database objects. The main DDL statements in SQL are:

1. \*\*CREATE\*\*: Used to create new database objects such as tables, indexes, views, and schemas.

- \*\*Create Table\*\*: Creates a new table with specified columns and constraints.

```sql

CREATE TABLE table\_name (

column1 datatype constraint,

column2 datatype constraint,

...

);

```

- \*\*Create Index\*\*: Creates an index on one or more columns of a table to improve query performance.

```sql

CREATE INDEX index\_name ON table\_name (column1, column2, ...);

```

- \*\*Create View\*\*: Creates a virtual table derived from one or more tables or views.

```sql

CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

```

2. \*\*ALTER\*\*: Used to modify the structure of existing database objects.

- \*\*Alter Table\*\*: Modifies the structure of an existing table (e.g., adding or dropping columns, modifying column data types, adding constraints).

```sql

ALTER TABLE table\_name

ADD column\_name datatype constraint;

```

3. \*\*DROP\*\*: Used to remove existing database objects.

- \*\*Drop Table\*\*: Deletes a table and all its data from the database.

```sql

DROP TABLE table\_name;

```

- \*\*Drop Index\*\*: Removes an index from the database.

```sql

DROP INDEX index\_name;

```

- \*\*Drop View\*\*: Deletes a view from the database.

```sql

DROP VIEW view\_name;

```

4. \*\*TRUNCATE\*\*: Removes all records from a table, but keeps the table structure intact.

```sql

TRUNCATE TABLE table\_name;

```

These statements are essential for defining and managing the structure of database objects in SQL databases. They allow you to create, modify, and delete tables, indexes, views, and other database entities according to your application's requirements.

DQL statement

DQL stands for Data Query Language in SQL, which is used to retrieve data from a database. The primary DQL statement in SQL is the `SELECT` statement. Here's a basic structure of a `SELECT` statement:

```sql

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

```

- `SELECT`: Specifies the columns you want to retrieve.

- `FROM`: Specifies the table or tables from which to retrieve the data.

- `WHERE` (optional): Specifies conditions that must be met for the rows to be selected. It filters the rows based on a specific criterion.

Example:

```sql

SELECT \*

FROM employees

WHERE department = 'IT';

```

In this example:

- `SELECT \*` retrieves all columns from the `employees` table.

- `FROM employees` specifies the table from which to retrieve the data (`employees`).

- `WHERE department = 'IT'` filters the rows to only those where the `department` column has the value 'IT'.

Other clauses and keywords can be added to the `SELECT` statement to enhance its functionality, such as `ORDER BY`, `GROUP BY`, `HAVING`, etc., depending on the specific requirements of the query.

Normalization

Normalization in the context of SQL databases refers to the process of organizing data to minimize redundancy and dependency by dividing large tables into smaller tables and defining relationships between them. This is crucial for ensuring data integrity and efficient querying. There are several normal forms (NF) defined to guide this process, with each subsequent normal form building upon the previous ones. The most commonly discussed normal forms are:

1. \*\*First Normal Form (1NF)\*\*:

- Eliminate repeating groups within rows.

- Create a separate table for each set of related data.

- Identify each set of related data with a primary key.

2. \*\*Second Normal Form (2NF)\*\*:

- Meet all requirements of 1NF.

- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.

- Create relationships between these new tables and their original tables using foreign keys.

3. \*\*Third Normal Form (3NF)\*\*:

- Meet all requirements of 2NF.

- Remove columns that are not dependent upon the primary key.

The process of normalization typically involves decomposing large tables into smaller, related tables to reduce redundancy and dependency. This leads to more efficient storage and querying of data, as well as improving data integrity by reducing the likelihood of anomalies such as update anomalies, insertion anomalies, and deletion anomalies.

### Example:

Consider a denormalized table `Employee` with columns `EmployeeID`, `EmployeeName`, `Department`, and `ManagerName`. This table could be normalized into two tables:

\*\*Employees Table\*\*:

- `EmployeeID` (Primary Key)

- `EmployeeName`

- `DepartmentID` (Foreign Key referencing Departments table)

\*\*Departments Table\*\*:

- `DepartmentID` (Primary Key)

- `DepartmentName`

- `ManagerName`

In this normalized structure:

- The `Employees` table contains only information directly related to employees.

- The `Departments` table contains department-specific information, including the manager's name.

This separation reduces redundancy (e.g., if multiple employees are in the same department, the department's details are stored once) and improves data integrity and maintainability.

DCL

In SQL, DCL (Data Control Language) statements are used to control access to data within a database. There are primarily two main DCL statements:

1. \*\*GRANT\*\*: This statement is used to give specific privileges or permissions to database users. Privileges could include the ability to SELECT, INSERT, UPDATE, DELETE, or even create tables or databases. For example:

```sql

GRANT SELECT, INSERT ON table\_name TO user\_name;

```

This grants the user `user\_name` the ability to SELECT rows from `table\_name` and INSERT new rows into it.

2. \*\*REVOKE\*\*: This statement is used to take away privileges previously granted to users. It can revoke specific privileges or all privileges from a user. For example:

```sql

REVOKE INSERT ON table\_name FROM user\_name;

```

This revokes the INSERT privilege from `user\_name` for `table\_name`.

DCL statements are critical in managing security within a database environment, ensuring that users have appropriate access rights and permissions to perform their tasks while safeguarding sensitive data from unauthorized access or modification.