Basics of Database Management System

Overview of DBMS

A Database Management System (DBMS) is a software that helps manage and organize data in a structured way. It serves as an intermediary between users and the database, providing an efficient and secure way to store, retrieve, and manipulate data.

- **Data Organization:**
  - DBMS organizes data into tables, which are like digital spreadsheets. Each table has rows and columns, and each row represents a record while each column represents a specific attribute.
- **Data Retrieval:**
  - Users can easily retrieve specific information from the database using queries. A query is a command that instructs the DBMS to fetch and display data meeting certain conditions.
- **Data Integrity:**
  - DBMS ensures data integrity by enforcing rules and constraints. It prevents the entry of invalid or inconsistent data, maintaining the accuracy and reliability of the information stored in the database.
- **Data Security:**
  - DBMS provides security features to control access to the database. Users can have different levels of permissions, ensuring that only authorized individuals can view or modify specific data.
- **Concurrency Control:**
  - In multi-user environments, where multiple people might be accessing and modifying data simultaneously, DBMS ensures that transactions occur in a controlled manner to prevent conflicts and maintain data consistency.
- **Data Independence:**
  - DBMS provides a layer of abstraction between the physical storage of data and the way users perceive it. This allows changes in the database structure (such as adding or removing fields) without affecting the application programs that use the data.
- **Backup and Recovery:**
  - DBMS facilitates data backup and recovery mechanisms. Regular backups can be created to prevent data loss due to hardware failures, accidental deletions, or other unforeseen events.

- **Scalability:**
    - DBMS is designed to handle the growth of data over time. It provides mechanisms for scaling up the system to accommodate increasing amounts of information without significant disruption.

MySQL

MySQL is a popular open-source Relational Database Management System (RDBMS) that is widely used for managing and storing structured data.

- **Open Source**
- **Relational Database Management System (RDBMS):**
    - MySQL follows the relational model, organizing data into tables with rows and columns. This structured approach makes it suitable for handling complex relationships and large datasets.
- **Structured Query Language (SQL):**
    - MySQL uses SQL as its query language. Users can interact with the database using SQL to perform operations such as inserting, updating, retrieving, and deleting data.
- **Multi-Platform Support:**
    - MySQL is compatible with various operating systems, including Windows, Linux, macOS, and others. This versatility allows developers to deploy MySQL on different platforms based on their requirements.
- **Scalability:**
    - MySQL is known for its scalability, making it suitable for both small-scale applications and large, enterprise-level systems. It supports vertical scaling (adding resources to a single server) as well as horizontal scaling (distributed databases and replication).
- **Performance Optimization:**
    - MySQL provides various features and tools for performance optimization, including indexing, caching mechanisms, and query optimization. These help enhance the speed and efficiency of database operations.
- **Community and Support:**
    - MySQL has a large and active community of developers and users. This community support includes forums, documentation, and contributed plugins, making it easier for users to find solutions to common issues.
- **Security:**
    - MySQL incorporates security features to protect data integrity and confidentiality. This includes user authentication, access control, and encryption mechanisms to secure sensitive information.
- **Data Replication:**
    - MySQL supports replication, allowing the creation of copies of the database for backup, load balancing, or fault tolerance. This feature is particularly useful in ensuring data availability and disaster recovery.

Overview of Workbench

MySQL Workbench is a visual tool designed for database architects, developers, and database administrators to interact with MySQL databases.

- **Database Design:**
  - MySQL Workbench allows users to visually design, model, and create MySQL databases. It provides a graphical interface for designing tables, defining relationships, and establishing data types.
- **SQL Development:**
  - The tool includes a SQL editor for writing and executing SQL queries, stored procedures, and scripts. It supports syntax highlighting, auto-completion, and other features to streamline SQL development.
- **Data Modeling:**
  - Users can create and edit entity-relationship diagrams (ERD) to model the structure of their databases. This visual representation helps in understanding and designing complex database schemas.
- **Server Administration:**
  - MySQL Workbench provides tools for managing MySQL server instances. Users can start and stop servers, configure server settings, and perform other administrative tasks through the graphical interface.
- **Database Migration:**
  - It supports database migration tools that allow users to migrate data from other database systems to MySQL. This can simplify the process of transitioning to MySQL from other platforms.
- **Backup and Recovery:**
  - MySQL Workbench facilitates the creation of database backups and the execution of recovery operations. This is crucial for ensuring data integrity and providing a safety net against data loss.
- **Cross-Platform Compatibility:**
  - MySQL Workbench is available for various operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of users.
- **Integration with MySQL Services:**
  - It integrates with MySQL services and features, such as MySQL Shell, enabling users to run advanced scripts and manage their MySQL instances efficiently.

Database Design

- Database design is the process of creating a blueprint that outlines how data will be organized, stored, and accessed within a database system.
- It involves defining the structure of the database, specifying the relationships between data elements, and establishing rules and constraints to ensure data integrity.
- The goal of database design is to create an efficient, scalable, and maintainable system that meets the requirements of the intended applications and users.
- A well-designed database ensures data integrity, accuracy, and efficiency in retrieving and manipulating information.

SQL Basics

Structured Query Language (SQL) is a standard programming language used for managing and manipulating relational databases. It provides a set of commands to interact with databases, enabling users to perform various operations such as retrieving, updating, inserting, and deleting data.

- **CREATE TABLE**: Defines a new table.

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ...
);
```

- **ALTER TABLE**: Modifies an existing table structure.

```
ALTER TABLE table_name
ADD column_name datatype;
```

- **DROP TABLE**: Deletes a table.

```
DROP TABLE table_name;
```

- **SELECT**: Retrieves data from one or more tables.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- **INSERT INTO**: Adds new records to a table.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

- **UPDATE**: Modifies existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- **DELETE FROM**: Removes records from a table.

```
DELETE FROM table_name
WHERE condition;
```

Data Sorting

Data sorting is the process of arranging data in a specific order, typically based on the values in one or more columns or fields. Sorting can be performed in ascending (from the smallest to the largest) or descending (from the largest to the smallest) order. Sorting is a fundamental operation in data processing, providing a structured way to analyze and present information.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```

Null Value & Keyword

1. NULL as a Value:

Used when inserting data to represent missing or undefined information.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, NULL, ...);
```

2. NULL as a Keyword:

Used in queries to filter rows where a specific column has a NULL value.

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IS NULL;
```

Reserved Keywords

A reserved keyword is a term that has a special meaning in the SQL language and cannot be used as an identifier (such as table or column names) unless enclosed in backticks, double quotes, or square brackets depending on the specific database system.

- **SELECT**: Retrieves data from one or more tables.
- **FROM**: Specifies the table or tables from which to retrieve data.
- **WHERE**: Filters the rows based on a specified condition.
- **AND**, **OR**, **NOT**: Logical operators used in conjunction with the WHERE clause.
- **INSERT**, **INTO**, **VALUES**: Adds new records to a table.
- **UPDATE**, **SET**: Modifies existing records in a table.
- **DELETE, FROM**: Removes records from a table.
- **CREATE**, **TABLE**, **INDEX**, **VIEW**, **PROCEDURE**, **FUNCTION**: Defines database objects.
- **ALTER**, **DROP**, **CONSTRAINT**: Modifies or deletes existing database objects.
- **JOIN**, **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, **FULL JOIN**: Combines rows from two or more tables based on related columns.
- **GROUP BY**: Groups rows based on specified columns.
- **HAVING**: Filters grouped rows based on a specified condition.
- **ORDER BY**, **ASC**, **DESC**: Sorts the result set based on one or more columns.
- **DISTINCT**: Retrieves unique values in a column.
- **AS**: Renames a column or table using an alias.
- **IN**, **BETWEEN**, **LIKE**, **IS NULL**: Operators used in WHERE clauses for specific conditions.
- **CASE**, **WHEN**, **THEN**, **ELSE**, **END**: Conditional statement for custom expressions.
- **UNION**, **INTERSECT**, **EXCEPT**: Combines or compares the result sets of two or more SELECT statements.
- **COUNT**, **SUM**, **AVG**, **MIN**, **MAX**: Aggregate functions for calculations on columns.
- **DEFAULT**: Provides a default value for a column.
- **INDEX**: Creates an index on one or more columns for improved query performance.
- **NOT NULL**: Specifies that a column cannot contain NULL values.
- **CHECK**: Adds a condition to a column, limiting the range of allowed values.

Auto increment

Auto-increment is a feature in SQL that allows a column, typically used for primary keys, to automatically generate unique values as new records are inserted into a table. This is commonly employed to ensure that each record in a table has a unique identifier without requiring explicit user input or manual assignment.

```
CREATE TABLE table_name (
    column_name INT AUTO_INCREMENT PRIMARY KEY,
    other_columns datatype,
    ...
);
```

Or

```
CREATE TABLE table_name (
    column_name INT IDENTITY(1,1) PRIMARY KEY,
    other_columns datatype,
    ...
);
```

- To specify that the "id" column should start at value 10 and increment by 5, change the autoincrement to `AUTOINCREMENT(10,5)`.

DDL, DML, DCL, TCL, DQL

1. **Data Definition Language (DDL):**
   - DDL statements are used to define the structure of the database.Common DDL commands include:
     - CREATE TABLE: Defines a new table.

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype, …
);
```

     - ALTER TABLE: Modifies an existing table structure.

```
ALTER TABLE table_name
ADD column_name datatype;
```

     - DROP TABLE: Deletes a table.

```
DROP TABLE table_name;
```

2. **Data Manipulation Language (DML):**
   - DML statements are used to manipulate data stored in the database. Common DML commands include:
     - SELECT: Retrieves data from one or more tables.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column1, column2, ...
HAVING condition
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```

     - INSERT INTO: Adds new records to a table.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

     - UPDATE: Modifies existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

     - DELETE FROM: Removes records from a table.

```
DELETE FROM table_name
WHERE condition;
```

3. **Data Control Language (DCL):**
    - DCL statements control access to data within the database. Common DCL commands include:
        - GRANT: Provides specific privileges to database users.

```
GRANT permission(s) ON object TO user/role;
```

        - REVOKE: Removes privileges from database users.

```
REVOKE permission(s) ON object FROM user/role;
```

4. **Transaction Control Language (TCL):**
    - TCL commands are responsible for managing transactions in a database, ensuring the integrity and consistency of data modifications. Common TCL commands include:
        - COMMIT: Commits the current transaction, making changes permanent in the database.

```
COMMIT;
```

        - ROLLBACK: Rolls back the current transaction, undoing changes made during the transaction, restoring the database to its state before the transaction started.

```
ROLLBACK;
```

5. **Data Query Language (DQL):**
    - DQL is a subset of SQL focused on querying data. The primary DQL command is SELECT, which retrieves data from one or more tables based on specified conditions.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Limit

In SQL, the LIMIT clause is used to restrict the number of rows returned by a query. It is often used in conjunction with the SELECT statement to limit the result set to a specified number of rows.

```
SELECT column1, column2, ...
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, ...]
LIMIT number_of_rows;
```

Aggregate functions

Aggregate functions in SQL are used to perform calculations on a set of values and return a single value.

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

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

- **SUM()**: Calculates the sum of values in a numeric column.

```
SELECT SUM(column_name)
 FROM table_name
WHERE condition;
```

- **AVG()**: Calculates the average of values in a numeric column.

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

- **MIN()**: Finds the minimum value in a column.

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

- **MAX()**: Finds the maximum value in a column.

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

Sub- Queries

A subquery, also known as an inner query or nested query, is a query embedded within another query. It allows you to retrieve data from one or more tables and use that result set as a condition or value in the main query. Subqueries are enclosed in parentheses and can appear in various parts of a SQL statement, such as the SELECT, FROM, or WHERE clauses.

```
SELECT column1, column2, …
FROM table1
WHERE column_name operator (SELECT column_name
                            FROM table2
                            WHERE condition);
```