DataBase

DBMS Full Form and Overview:
The full form of DBMS is Database Management System.
It is software that provides the functionality to organize, store, manage, and access data.
Its main purpose is to handle data efficiently and prevent unauthorized access.

Components of DBMS

Database: A collection of data organized in tables, rows, and columns.

DBMS Software: Software used to manage data (e.g., MySQL, Oracle, SQL Server).

User: The person who uses or manages the data.

Query Language: Commands that allow interaction with the database (e.g., SQL)

Features of DBMS

Data Redundancy Control: Prevents the same data from being stored in multiple places.

Data Security: Protects data from unauthorized access.

Data Consistency: Ensures one consistent version of data is available to all users.

Backup and Recovery: Provides systems to recover data in case of loss.

Concurrent Access: Allows multiple users to access data simultaneously.

Popular DBMS Software

MySQL

PostgreSQL

Oracle Database

Microsoft SQL Server

MongoDB (NoSQL)

Introduction to MySQL

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and manipulate data.

Key Features of MySQL

Open-Source

Relational Database

Cross-Platform

High Performance

Scalability

Security

Multi-User Access

introduction to workbench

MySQL Workbench is a GUI (Graphical User Interface) tool used to interact with databases. It is used to design, manage, and query MySQL databases.

MySQL Workbench Interface

Home Screen: Access your saved connections from here.

SQL Editor: Write and execute queries.

Data Modeling Tool: Create ER diagrams.

Advantages of MySQL Workbench

User-Friendly: The visual interface is highly intuitive.

All-in-One Tool: Combines designing, querying, and administration in one place.

Cross-Platform: Available on Windows, macOS, and Linux.

What is Database Design?

Database design is the process of creating a structured framework for how data will be stored, organized, and managed in a database

Key Components of Database Design

Entities

Attributes

Relationships

Primary Key

Foreign Key

Constraints

Example of a Simple Database Design

Entities and Tables:

1. **Users Table**
    - `UserID` (Primary Key)
    - `Name`
    - `Email`
2. **Products Table**
    - `ProductID` (Primary Key)
    - `Name`
    - `Price`
3. **Orders Table**
    - `OrderID` (Primary Key)
    - `UserID` (Foreign Key linking to Users)
    - `ProductID` (Foreign Key linking to Products)
    - `Quantity`

Benefits of Good Database Design

1. Data Integrity
2. Efficiency
3. Scalability
4. Ease of Maintenance

What is SQL?

SQL (Structured Query Language) is a standard language used to communicate with and manage databases. It allows you to create, read, update, and delete data stored in a database. SQL works with relational databases like MySQL, PostgreSQL, Oracle, and SQL Server.

Basic SQL Commands

SQL commands are categorized into four main types:

1. **DDL (Data Definition Language)**: Used to define or modify the structure of a database.
    - `CREATE`: Create a new database or table.
    - `ALTER`: Modify an existing table.
    - `DROP`: Delete a table or database.
2. **DML (Data Manipulation Language)**: Used to manipulate data in the database.
    - `INSERT`: Add new records.
    - `UPDATE`: Modify existing records.
    - `DELETE`: Remove records.
3. **DQL (Data Query Language)**: Used to query or fetch data from the database.
    - `SELECT`: Retrieve data from tables.
4. **DCL (Data Control Language)**: Used to control access to data.
    - `GRANT`: Give access to users.
    - `REVOKE`: Remove access from users.

Data Sorting in SQL

Sorting means arranging data in a specific order (ascending or descending) based on one or more columns. In SQL, sorting is done using the `ORDER BY` clause.

`ASC`: Ascending order (default). Smallest to largest for numbers, A to Z for text.

`DESC`: Descending order. Largest to smallest for numbers, Z to A for text.

Sorting with NULL Values

By default, NULL values are sorted first in ascending order and last in descending order.

What is AUTO_INCREMENT in SQL?

The `AUTO_INCREMENT` feature in SQL is used to automatically generate a unique value for a column (usually the primary key) whenever a new record is inserted into a table.

Key Features of AUTO_INCREMENT

1. Automatic Value Assignment
2. Unique Values
3. Incremental
4. Starts from 1

Limitations of AUTO_INCREMENT

Gap in Numbers

SQL Commands Classification

SQL commands are categorized into five main types based on their purpose: DCL, DDL, DML, TCL, and DQL.

1. DCL (Data Control Language)

DCL commands manage permissions and access control to the database.

Commands:

`GRANT`: Gives specific permissions to a user.

```
GRANT SELECT, INSERT ON Employees TO 'username';
```

`REVOKE`: Removes permissions from a user.

```
REVOKE SELECT, INSERT ON Employees FROM 'username';
```

Use Case:

Control who can view or modify the database.

2. DDL (Data Definition Language)

DDL commands are used to define and modify the structure of the database.

Commands:

CREATE: Creates a new database, table, or object.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT
);
```

ALTER: Modifies an existing table's structure.

```
ALTER TABLE Employees ADD Salary INT;
```

DROP: Deletes a table or database.

```
DROP TABLE Employees;
```

TRUNCATE: Removes all data from a table but keeps the structure.

```
TRUNCATE TABLE Employees;
```

Use Case:

Manage the design and structure of the database.

3. DML (Data Manipulation Language)

DML commands handle data manipulation (insert, update, delete) in the database.

Commands:

`INSERT`: Adds new records to a table.

```
INSERT INTO Employees (EmployeeID, Name, Age) VALUES
(1, 'John', 30);
```

`UPDATE`: Modifies existing records.

```
UPDATE Employees SET Age = 31 WHERE EmployeeID = 1;
```

`DELETE`: Removes specific records from a table.

```
DELETE FROM Employees WHERE EmployeeID = 1;
```

Use Case:

Work with the data stored in the database.

4. TCL (Transaction Control Language)

TCL commands manage transactions in the database to ensure consistency and reliability.

Commands:

`COMMIT`: Saves all changes made during a transaction.
```
COMMIT;
```

`ROLLBACK`: Reverts changes made during a transaction.

```
ROLLBACK;
```

`SAVEPOINT`: Creates a checkpoint within a transaction to roll back to a specific point.
```
SAVEPOINT Save1;
```

`SET TRANSACTION`: Sets the properties of a transaction.
`SET TRANSACTION READ ONLY;`

Use Case:

Maintain database integrity during operations like banking transactions.

5. DQL (Data Query Language)

DQL commands are used to retrieve data from the database.

Commands:

`SELECT`: Retrieves data from one or more tables.

```
SELECT * FROM Employees;
```

`WHERE`: Filters records based on conditions.
`SELECT Name, Age FROM Employees WHERE Age > 30;`

Use Case:

Fetch and display data for analysis or reporting.

The `LIMIT` Keyword in SQL

The `LIMIT` keyword is used to restrict the number of rows returned by a `SELECT` query

SELECT column1, column2, ... FROM table_name LIMIT number_of_rows;

Aggregate Functions in SQL

Aggregate functions are special SQL functions that perform calculations on a group of values and return a single result. These are commonly used in data analysis and reporting.

List of Aggregate Functions

1. `COUNT()`: Counts the number of rows.
2. `SUM()`: Calculates the total sum of a column.
3. `AVG()`: Calculates the average (mean) of a column.
4. `MIN()`: Finds the minimum value in a column.
5. `MAX()`: Finds the maximum value in a column.

Subqueries in SQL

A subquery is a query nested inside another query. Subqueries are used to perform intermediate calculations or fetch data that the main query depends on.

Types of Subqueries

1. Single-row subqueries: Return only one row of data.
2. Multi-row subqueries: Return multiple rows of data.
3. Scalar subqueries: Return a single value.
4. Correlated subqueries: Refer to columns from the outer query and are evaluated for each row of the outer query.

Joins in SQL

Joins in SQL are used to combine rows from two or more tables based on a related column. They allow you to retrieve meaningful data by linking tables that have relationships.

Types of Joins

1. **INNER JOIN**: Retrieves rows with matching values in both tables.
2. **LEFT JOIN (LEFT OUTER JOIN)**: Retrieves all rows from the left table and matching rows from the right table. If no match, returns `NULL` for the right table's columns.
3. **RIGHT JOIN (RIGHT OUTER JOIN)**: Retrieves all rows from the right table and matching rows from the left table. If no match, returns `NULL` for the left table's columns.
4. **FULL JOIN (FULL OUTER JOIN)**: Retrieves all rows when there is a match in either table, and fills unmatched rows with `NULL`.
5. **CROSS JOIN**: Produces a Cartesian product (all possible combinations of rows).
6. **SELF JOIN**: Joins a table to itself.

UNION in SQL

The UNION operator is used to combine the results of two or more `SELECT` queries into a single result set.

By default, it removes duplicate rows. All queries involved in a `UNION` must have the same number of columns, with matching data types in corresponding columns.

Syntax

sql

Copy code

```
SELECT column1, column2, ...

FROM table1

WHERE condition
```

```
UNION

SELECT column1, column2, ...

FROM table2

WHERE condition;
```

Rules for Using UNION

1. **Same Number of Columns**: Each `SELECT` query must have the same number of columns.
2. **Matching Data Types**: The data types of corresponding columns must be compatible.
3. **Column Order**: The columns in each `SELECT` statement must be in the same order.

Index in SQL

An **index** in SQL is a database object that improves the speed of data retrieval operations on a table.

Why Use an Index?

Without an index, the database has to scan every row in the table (called a **full table scan**) to find the desired data.

With an index, the database can quickly locate the data, significantly improving query performance.

Syntax to Create an Index

1. Creating a Simple Index

```
CREATE INDEX index_name

ON table_name (column_name);
```

DROP INDEX index_name ON table_name;

Advantages of Indexing

1. **Faster Data Retrieval**: Reduces the time required for SELECT queries.
2. **Unique Constraint**: Unique indexes ensure data integrity.
3. **Sorting**: Helps in sorting results faster.

View in SQL

A view in SQL is a virtual table that is based on the result of a SELECT query.

It does not store data itself but provides a way to access and display data from one or more tables.

Why Use a View?

1. **Simplify Complex Queries**: Save a complicated query and reuse it as a virtual table.
2. **Enhance Security**: Restrict access to specific columns or rows by exposing only selected data.
3. **Data Abstraction**: Provide a specific view of the data without exposing the underlying table structure.
4. **Reusable Code**: Save time by using predefined views in multiple queries.

Advantages of Views

1. **Simplifies Queries**: Makes it easier to work with complex queries.
2. **Data Security**: Restricts access to sensitive data by exposing only specific parts of a table.
3. **Logical Independence**: Changes in the underlying table do not affect the view structure.
4. **Reusable**: Can be used in multiple queries like a table.

Backup and Restore in SQL

1. Backup

Backup is the process of creating a copy of your database to ensure data recovery in case of data loss, corruption, or failure.

- A backup contains all the data, schemas, and database structures.

Syntax for Backing Up a Database:

```
mysqldump -u [username] -p [database_name] >
[backup_file.sql]
```

2. Restore

Restore is the process of reloading a database from a backup file.

Syntax for Restoring a Database:

```
mysql -u [username] -p [database_name] < [backup_file.sql]
```

EXPLAIN Keyword in SQL

The EXPLAIN keyword is used to analyze and understand how a SQL query is executed by the database. It provides details like the order of execution, indexes used, and the estimated cost of the query.

Why Use EXPLAIN?

1. **Optimize Queries**: Helps identify slow queries and improve performance.
2. **Understand Query Execution Plan**: Reveals how tables are joined, scanned, and filtered.
3. **Detect Inefficient Index Usage**: Shows whether indexes are used effectively.

Syntax

```
EXPLAIN SELECT column1, column2

FROM table_name

WHERE condition;
```