Introduction

Purpose of MySQL

MySQL is designed to store, manage, and retrieve structured data.

It provides a reliable and scalable solution for various applications, such as websites, e-commerce systems, and enterprise-level software.

Basic Concepts of MySQL

- Introduce the following fundamental concepts:

- Databases: Containers for organizing related data.

- Sample Query: CREATE DATABASE mydatabase;

- Tables: Structures within databases to store data in rows and columns.

- Sample Query: CREATE TABLE employees (id INT PRIMARY KEY, name VARCHAR(50), age INT, department VARCHAR(50));

- Rows: Individual records in a table.

- Columns: Fields or attributes that define the type of data stored in a table.

- Sample Query: INSERT INTO employees (name, age, department) VALUES ('John Smith', 30, 'Marketing');

- Primary Key: A unique identifier for each row in a table.

- Foreign Key: A field that establishes a relationship between two tables.

SQL Basics

- SELECT: Retrieves data from one or more tables.

- Sample Query: SELECT \* FROM employees;

- INSERT: Adds new records into a table.

- Sample Query: INSERT INTO employees (name, age, department) VALUES ('Jane Doe', 25, 'Sales');

- UPDATE: Modifies existing records in a table.

- Sample Query: UPDATE employees SET age = 31 WHERE id = 1;

- DELETE: Removes records from a table.

- Sample Query: DELETE FROM employees WHERE id = 1;

- CREATE: Creates new databases, tables, or other database objects.

- Sample Query: CREATE TABLE customers (id INT PRIMARY KEY, name VARCHAR(50));

- ALTER: Modifies the structure of a database object.

- Sample Query: ALTER TABLE employees ADD COLUMN email VARCHAR(50);

- DROP: Deletes databases, tables, or other database objects.

- Sample Query: DROP TABLE customers;

Retrieving Data with SELECT

Explain the SELECT statement in detail, including:

- Syntax: SELECT column1, column2 FROM table\_name WHERE condition;

- Retrieving specific columns or all columns using "\*".

- Filtering data using the WHERE clause.

- Sample Query: SELECT name, age FROM employees WHERE department = 'Marketing';

- Sorting data with the ORDER BY clause.

- Sample Query: SELECT \* FROM employees ORDER BY age DESC;

- Limiting the number of rows returned using the LIMIT clause.

- Sample Query: SELECT \* FROM employees LIMIT 10;

Inserting, Updating, and Deleting Data

- Title: Inserting, Updating, and Deleting Data

- Explain the INSERT, UPDATE, and DELETE statements, including:

- INSERT: Syntax, specifying columns and values.

- Sample Query: INSERT INTO employees (name, age, department) VALUES ('John Doe', 28, 'Finance');

- UPDATE: Syntax, modifying specific columns with new values.

- Sample Query: UPDATE employees SET age = 29 WHERE id = 2;

- DELETE: Syntax, removing rows based on specified conditions.

- Sample Query: DELETE FROM employees WHERE id = 3;

Creating and Modifying Database Structures

- Title: Creating and Modifying Database Structures

- Explain the CREATE and ALTER statements, including:

- CREATE DATABASE: Syntax for creating a new database.

- Sample Query: CREATE DATABASE mydatabase;

- CREATE TABLE: Syntax for creating a new table with columns and data types.

- Sample Query: CREATE TABLE customers (id INT PRIMARY KEY, name VARCHAR(50));

- ALTER TABLE: Syntax for modifying existing tables (e.g., adding or dropping columns).

- Sample Query: ALTER TABLE employees ADD COLUMN email VARCHAR(50);

In SQL (Structured Query Language), there are several keywords that are used to perform different types of operations on a database. Here are some key SQL keywords and their functionalities:

1. DDL (Data Definition Language): DDL statements are used to define and manage the structure of database objects, such as tables, views, indexes, and schemas. Some common DDL keywords include:

- CREATE: Used to create a new database object, such as a table or view.

- ALTER: Used to modify the structure of an existing database object.

- DROP: Used to delete a database object.

- TRUNCATE: Used to remove all data from a table.

2. DML (Data Manipulation Language): DML statements are used to manipulate and retrieve data within database objects. These statements affect the data within tables. Some common DML keywords include:

- SELECT: Used to retrieve data from one or more tables.

- INSERT: Used to insert new rows of data into a table.

- UPDATE: Used to modify existing data within a table.

- DELETE: Used to delete specific rows of data from a table.

3. DCL (Data Control Language): DCL statements are used to control the access and permissions of database objects. These statements are typically used by database administrators. Some common DCL keywords include:

- GRANT: Used to grant specific privileges and permissions to users or roles.

- REVOKE: Used to revoke previously granted privileges and permissions.

4. TCL (Transaction Control Language): TCL statements are used to manage the changes made to a database within a transaction. These statements ensure data integrity and control the flow of transactions. Some common TCL keywords include:

- COMMIT: Used to permanently save the changes made in a transaction.

- ROLLBACK: Used to undo or cancel the changes made in a transaction.

- SAVEPOINT: Used to set a savepoint within a transaction.

ACID stands for Atomicity, Consistency, Isolation, and Durability. It is a set of properties that ensure reliability and integrity in database transactions.

- Atomicity: All the operations in a transaction are treated as a single unit of work. Either all the operations are executed successfully, or none of them are. For example, transferring funds from one bank account to another is an atomic transaction. If any step fails, the entire transaction is rolled back, ensuring data consistency.

- Consistency: The database remains in a consistent state before and after the transaction. Constraints and rules defined in the database schema are enforced during the transaction. For example, if a foreign key constraint is defined between two tables, the database ensures that a valid value is assigned to the foreign key column.

- Isolation: Each transaction operates independently of other transactions. Transactions are executed in isolation, preventing interference and ensuring that each transaction sees a consistent snapshot of the database. For example, if two transactions try to update the same data simultaneously, isolation ensures that they do not interfere with each other and produce inconsistent results.

- Durability: Once a transaction is committed, the changes made to the database are permanent and will survive any subsequent system failures. The data is stored securely and can be recovered in case of failures. For example, after a transaction is successfully completed, even if the system crashes, the changes made by the transaction will be persisted.

In summary, ACID properties ensure that database transactions are reliable, consistent, isolated, and durable, providing a robust foundation for data integrity and transaction management in MySQL.

Types of Joins

- Title: Types of Joins

- Introduce the following types of joins commonly used in MySQL:

- Inner Join: Returns only the matching rows from both tables.

- Left Join: Returns all rows from the left (or first) table and the matching rows from the right (or second) table.

- Right Join: Returns all rows from the right (or second) table and the matching rows from the left (or first) table.

- Full Outer Join: Returns all rows from both tables, including the unmatched rows from either table.

- Title: Inner Join

- Explain that an inner join returns only the matching rows from both tables based on the common column.

- Provide an example query using the INNER JOIN keyword:

- Sample Query: SELECT employees.name, departments.department\_name FROM employees INNER JOIN departments ON employees.department\_id = departments.department\_id;

- Title: Left Join

- Explain that a left join returns all rows from the left table and the matching rows from the right table.

- Provide an example query using the LEFT JOIN keyword:

- Sample Query: SELECT employees.name, departments.department\_name FROM employees LEFT JOIN departments ON employees.department\_id = departments.department\_id;

- Title: Right Join

- Explain that a right join returns all rows from the right table and the matching rows from the left table.

- Provide an example query using the RIGHT JOIN keyword:

- Sample Query: SELECT employees.name, departments.department\_name FROM employees RIGHT JOIN departments ON employees.department\_id = departments.department\_id;

- Title: Full Outer Join

- Explain that a full outer join returns all rows from both tables, including the unmatched rows from either table.

- Mention that MySQL does not have a built-in syntax for a full outer join, but it can be achieved using other join types and UNION.

- Provide an example query combining LEFT JOIN and RIGHT JOIN using UNION:

- Sample Query: SELECT employees.name, departments.department\_name FROM employees LEFT JOIN departments ON employees.department\_id = departments.department\_id UNION SELECT employees.name, departments.department\_name FROM employees RIGHT JOIN departments ON employees.department\_id = departments.department\_id WHERE employees.department\_id IS NULL;

**TASK ->** [**https://sqlbolt.com/**](https://sqlbolt.com/) **🡪 COMPLETE B4 12.15PM**