DBMS (DATABASE MANAGEMENT SYSTEM)

Name : Ragini Singh
Cohort : Jensen Huang
Roll No: 150096724023
Topic : DBMS Lab Session
Date: 27 Jan 2025
Teacher Name : Sandhya Jawale

Question No: 1. What is DML?
Answer : DML refers to the subset of SQL commands used for managing data within database tables. It allows you to perform operations such as retrieving, inserting, updating, and deleting data.

Key Operations in DML:

- SELECT: Retrieves data from a database.
- INSERT: Adds new rows of data into a table.
- UPDATE: Modifies existing data within a table.
- DELETE: Removes data from a table.

Example :

```
-- Inserting a new record
INSERT INTO employees (id, name, department) VALUES (1, 'John Doe', 'Engineering');

-- Selecting all records
SELECT * FROM employees;

-- Updating a record
UPDATE employees SET department = 'Marketing' WHERE id = 1;

-- Deleting a record
DELETE FROM employees WHERE id = 1;
```

Question 2. What are the commands in DML?

DML Commands:

1. **SELECT**: Retrieves data from one or more tables.
   - Example: `SELECT * FROM employees;`

2. **INSERT**: Adds a new row of data into a table.
   - Example: `INSERT INTO employees (id, name, department) VALUES (2, 'Alice Smith', 'HR');`
3. **UPDATE**: Modifies existing data in a table.
   - Example: `UPDATE employees SET department = 'Sales' WHERE id = 2;`
4. **DELETE**: Removes one or more rows of data from a table.
   - Example: `DELETE FROM employees WHERE id = 2;`

Question 3. What is a Primary Key?

A **Primary Key** is a unique identifier for a record in a database table. It ensures that each record is distinct and prevents duplication.

Two Important Rules:

1. **Uniqueness**: Each primary key value must be unique.
2. **Non-nullability**: A primary key cannot have a NULL value.

Example :

```
CREATE TABLE employees (
   id INT PRIMARY KEY,
   name VARCHAR(100),
   department VARCHAR(100)
);
```

Question 4. What are the types of databases?

There are various types of databases, depending on how they organize and manage data. Here are three common types:

1. **Relational Databases (RDBMS)**: Organize data into tables (relations) with rows and columns.
   - Example: **MySQL**, **PostgreSQL**, **Oracle Database**.
2. **NoSQL Databases**: Handle unstructured or semi-structured data, often for large-scale, distributed systems.
   - Example: **MongoDB**, **Cassandra**, **CouchDB**.

3. **Hierarchical Databases**: Data is stored in a tree-like structure, with parent-child relationships.
    - Example: **IBM IMS**.

Question 5. What are the types of relationships in databases?

Types of Relationships:

1. **One-to-One**: One record in a table is related to one record in another table.
    - Example: A **Person** table with a **Passport** table, where each person has only one passport.

One-to-Many: One record in a table can relate to many records in another table.

- Example: A **Department** table and an **Employee** table, where one department can have multiple employees.

Many-to-Many: Many records in one table can relate to many records in another table.

- Example: A **Student** table and a **Course** table, where each student can enroll in multiple courses, and each course can have multiple students.

Question 6. What is an Entity?

An **Entity** in a database is any object, event, or concept about which data is stored. It can be a real-world object, such as a person or product, or an abstract concept.

Difference Between Entity & Attribute:

- **Entity**: Represents a thing or object in the database.
    - Example: **Employee**, **Customer**.
- **Attribute**: A property or characteristic of an entity.
    - Example: **Employee** entities might have attributes like `EmployeeID`, `Name`, `Salary`.

Real-Life Example:

- **Entity**: Car.
- **Attributes**: `Model`, `Color`, `Year`.

Question 7. What are the types of Inner and Outer Joins?

Types of Joins:

1. **Inner Join**: Returns records that have matching values in both tables.
    - Example:

```
SELECT employees.name, departments.name
FROM employees
INNER JOIN departments ON employees.department_id = departments.id;
```

Left Outer Join: Returns all records from the left table, and matched records from the right table. If no match, NULL values are returned for the right table.

- Example:
- SELECT employees.name, departments.name
- FROM employees
- RIGHT OUTER JOIN departments ON employees.department_id = departments.id;

Right Outer Join: Similar to the Left Outer Join, but returns all records from the right table and matched records from the left table.

- Example:

```
SELECT employees.name, departments.name
FROM employees
RIGHT OUTER JOIN departments ON employees.department_id = departments.id;
```

Full Outer Join: Returns all records when there is a match in either left or right table.

- Example:

```
SELECT employees.name, departments.name
FROM employees
FULL OUTER JOIN departments ON employees.department_id = departments.id;
```

Question 8. Difference between Primary Key and Unique Key?

Comparison:

1. **Uniqueness**:
    - Both Primary Key and Unique Key enforce uniqueness, but only **Primary Key** can have one constraint per table.
2. **Nullability**:
    - **Primary Key**: Cannot accept NULL values.
    - **Unique Key**: Can accept NULL values (but only one NULL).
3. **Purpose**:
    - **Primary Key**: Uniquely identifies a record in the table.
    - **Unique Key**: Ensures that all values in a column are unique.

SQL Example:

- **Primary Key**:

```
CREATE TABLE employees (
   id INT PRIMARY KEY,
   name VARCHAR(100)
);
```

Unique Key:

```
CREATE TABLE employees (
   id INT PRIMARY KEY,
   email VARCHAR(100) UNIQUE
);
```

Question 9. What is a Relationship in a Database Model?

A **Relationship** in a database model defines how tables are related to each other. These relationships are established using **Foreign Keys**.

Foreign Key Purpose: Foreign Keys enforce referential integrity by linking records in one table to records in another.

Example:

1. **Departments** table:

```
CREATE TABLE departments (
   id INT PRIMARY KEY,
   name VARCHAR(100)
);
```

Employees table (with Foreign Key):

```
CREATE TABLE employees (
   id INT PRIMARY KEY,
   name VARCHAR(100),
   department_id INT,
   FOREIGN KEY (department_id) REFERENCES departments(id)
);
```

Question 10. What is the Foreign Key?

A **Foreign Key** is a column (or a set of columns) in a table that uniquely identifies a row of another table. It is used to create and enforce a link between the data in two tables.

Purpose: A foreign key is used to maintain referential integrity by ensuring that the value in one table matches a value in another table.

SQL Example:

```
CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   order_date DATE,
   customer_id INT,
   FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```