**MODULE 5: DATABASE**

**Q.1 What do you understand by Database?**

A database is a structured collection of data organized for efficient storage, retrieval, and management. It uses tables to represent entities and columns for their attributes. Databases facilitate data organization, retrieval, and manipulation through a query language. Various types of databases exist, including relational and NoSQL databases, serving different data storage and retrieval needs in applications and systems.

**Q.2 What is Normalization?**

Normalization is the process of organizing a relational database to reduce redundancy and improve data integrity. It involves breaking down large tables, minimizing duplicated data, and establishing relationships between tables. This ensures efficient storage, reduces the likelihood of errors, and simplifies data maintenance. The goal is to strike a balance between avoiding redundancy and maintaining simplicity in database design.

**Q.3 What is difference between DBMS and RDBMS?**

DBMS (Database Management System) is a general term for software that manages databases, while RDBMS (Relational Database Management System) is a specific type of DBMS that organizes data using a relational model. RDBMS enforces relationships between tables, ensuring data integrity, and follows a structured approach, whereas DBMS can handle various data structures with less emphasis on a specific model. Examples of RDBMS include MySQL and PostgreSQL, while file systems and IMS are examples of more general-purpose DBMS.

**Q.4 What is Codd’s Rule of RDBMS Systems?**

Codd's Rules are a set of principles for relational database management systems (RDBMS). Here are a few key points:

1. Information Rule:

- Data should be stored in tables.

2. Guaranteed Access Rule:

- Each piece of data should be accessible using a table name, primary key, and column name.

3. Systematic Treatment of Null Values:

- The system should handle empty values.

4. Dynamic Online Catalog Based on The Relational Model:

- Database structure should be stored in a table, accessible using the same language as data.

5. Comprehensive Data Sublanguage Rule:

- The system must support a language for defining data, manipulating it, enforcing rules, and managing transactions.

6. View Updating Rule:

- If a view can be theoretically updated, the system should be able to update it.

These rules ensure consistency, integrity, and accessibility in relational databases.

**Q.5 What do you understand by Data Redundancy?**

Data redundancy happens when the same information is stored in multiple places in a database, leading to wasted storage and potential inconsistencies in the data.

**Q.6 What is DDL Interpreter?**

A DDL (Data Definition Language) interpreter is part of a database system that processes commands related to defining or modifying the structure of a database. It executes statements like `CREATE TABLE` or `ALTER TABLE` to create or change the database schema.

**Q.7 What is DML Compiler in SQL?**

A DML (Data Manipulation Language) compiler in SQL is a component of a database management system (DBMS) that processes and executes commands related to data manipulation. DML commands include operations such as inserting, updating, and deleting data in the database.

The DML compiler takes SQL statements like `INSERT`, `UPDATE`, and `DELETE` and translates them into low-level instructions that interact with the database to perform the specified data manipulation tasks. It is responsible for executing these commands efficiently and ensuring the integrity of the data.

In summary, a DML compiler in SQL is a key part of a DBMS that handles the execution of commands focused on manipulating the data stored in the database.

**Q.8 What are SQL Key Constraints? Explain with an example.**

1. Primary Key Constraint:

- Uniquely identifies each record.

- Example: CREATE TABLE Students (StudentID INT PRIMARY KEY, FirstName VARCHAR(50));

2. Unique Key Constraint:

- Ensures uniqueness but allows NULL values.

- Example: CREATE TABLE Employees (EmployeeID INT UNIQUE, FirstName VARCHAR(50));

3. Foreign Key Constraint:

- Links two tables, ensuring values match between them.

- Example:

CREATE TABLE Orders (OrderID INT PRIMARY KEY, ProductID INT, FOREIGN KEY (ProductID) REFERENCES Products(ProductID));

CREATE TABLE Products (ProductID INT PRIMARY KEY, ProductName VARCHAR(50));

**Q.9 What is save Point? How to create a save point? Provide example.**

A savepoint in a database allows you to set a point within a transaction to which you can later roll back if needed. It's a way to provide partial rollback functionality within a transaction.

Here's an example in SQL:

SAVEPOINT my\_savepoint;

UPDATE my\_table SET column1 = value1 WHERE condition;

ROLLBACK TO my\_savepoint;

In this example, `SAVEPOINT my\_savepoint;` creates a savepoint within the transaction. Later, you can use `ROLLBACK TO my\_savepoint;` to undo changes made after the savepoint. This is useful if part of the transaction encounters an issue and you want to roll back to a specific point rather than the beginning of the entire transaction.

**Q.10 What is trigger and how to create a trigger in SQL?**

A trigger in SQL is a set of instructions that are automatically executed (or "triggered") in response to certain events on a particular table or view. These events include INSERT, UPDATE, DELETE, and other data-related actions.

Here's a brief example of how to create a simple trigger:

CREATE TRIGGER my\_trigger

AFTER INSERT ON my\_table

FOR EACH ROW

BEGIN

INSERT INTO log\_table (event\_description) VALUES ('New row inserted');

END;

In this example:

- `my\_trigger` is the name of the trigger.

- `AFTER INSERT ON my\_table` specifies that the trigger will activate after an INSERT operation on `my\_table`.

- `FOR EACH ROW` indicates that the trigger operates on each row affected by the triggering statement.

The trigger logic, located between `BEGIN` and `END`, defines the actions to be taken when the trigger is activated. In this case, it inserts a log entry into `log\_table` whenever a new row is inserted into `my\_table`.