**Ketan Pillai**

**MODULE: 5 (Database)**

**Basics of Database**

**1. What do you understand By Database?**

Ans: A database is a structured collection of organized data that is typically stored electronically in a computer system. It allows for efficient storage, retrieval, and manipulation of data, making it easier to manage large volumes of information. Databases are designed to support data integrity, security, and concurrency control, ensuring that data remains accurate and accessible to users and applications. Common database systems include relational databases (like MySQL, PostgreSQL), NoSQL databases (like MongoDB, Redis), and others tailored for specific data types and use cases.

**2. What is Normalization?**

Ans: Normalization is a process used to organize data in a database efficiently by reducing redundancy and dependency. It involves breaking down large tables into smaller ones and defining relationships between them to improve data integrity and minimize anomalies during data manipulation. The goal is to ensure each piece of information is stored only once and to optimize the database structure for better performance and maintenance.

**3. What is Difference between DBMS and RDBMS?**

Ans: The main difference between DBMS (Database Management System) and RDBMS (Relational Database Management System) lies in their data storage and organization capabilities:

1. **DBMS (Database Management System):**

- A DBMS is a software system that manages databases. It allows users to define, create, and manage databases, providing tools for data storage, retrieval, modification, and deletion.

- DBMS can handle different types of data models, including hierarchical, network, and relational models.

- DBMS does not necessarily enforce relationships between tables or support complex querying capabilities like joins.

2. **RDBMS (Relational Database Management System):**

- RDBMS is a type of DBMS that organizes data into tables (relations) with predefined relationships between them using primary keys, foreign keys, and constraints.

- RDBMS enforces a tabular structure where data is stored in rows (tuples) and columns (attributes). Tables can be related to each other through defined relationships.

- RDBMS supports SQL (Structured Query Language) for querying and manipulating data, including powerful features like joins, aggregations, and transactions.

- Examples of RDBMS include MySQL, PostgreSQL, Oracle Database, SQL Server, etc.

**4. What is MF Cod Rule of RDBMS Systems?**

Ans: Codd's original 12 rules were intended to establish the theoretical foundation and requirements for relational database systems. Here is a summary of some of the key principles from Codd's rules:

1. **Information Rule**: All data should be stored in tables (relations) that are logically represented as n-ary relations.

2. **Guaranteed Access Rule**: Every value in a relational database is guaranteed to be accessible by using a combination of table name, primary key value, and column name.

3. **Systematic Treatment of Null Values**: Null values should be supported and treated in a systematic way that is independent of data type.

4. **Active Online Catalog**: The database's structure (metadata) should be stored in the database itself, allowing it to be queried and updated like any other data.

5. **Comprehensive Data Sublanguage Rule**: The DBMS must support a comprehensive language for defining, manipulating, and querying data.

6. **View Updating Rule**: Views (derived relations) that are theoretically updatable should be updatable by the system.

7. **High-level Insert, Update, and Delete**: The DBMS should support high-level insert, update, and delete operations that are consistent with the relational model and do not require users to write low-level code.

8. **Physical Data Independence**: Changes to the physical storage structures of the database (like indexes or storage devices) should not affect higher-level application programs.

9. **Logical Data Independence**: Changes to the logical structure (schema) of the database should not require changes to application programs.

10. **Integrity Independence**: The integrity constraints of the database (like primary key, foreign key constraints) should be stored in the catalog and not be directly coded into application programs.

11. **Distribution Independence**: The end user should not be able to see that the data is distributed over various locations.

12. **Non-subversion Rule**: If the system provides low-level access to the data, it should not be possible to bypass the integrity rules of the DBMS.

**5. What do you understand By Data Redundancy?**

Ans: Data redundancy refers to the situation where the same piece of data is stored in multiple places within a database or across multiple databases. This redundancy can occur intentionally or unintentionally due to poor database design or application architecture.

Data redundancy can lead to several issues:

1. **Wasted Storage Space:** Storing the same data multiple times consumes unnecessary storage space, which can increase costs and resource usage.

2. **Data Inconsistency:** When the same data is stored in multiple locations, there is a risk of inconsistencies arising if the data is updated in one place but not updated in another. This can lead to discrepancies and incorrect information being used in different parts of the system.

3. **Increased Complexity:** Managing redundant data adds complexity to the database design and application logic. It can make it harder to maintain and update the system.

4. **Decreased Data Integrity:** Redundant data can compromise data integrity, as updates or deletions of data may not be properly synchronized across all instances of the data.

**6. What is DDL Interpreter?**

Ans: DDL (Data Definition Language) interpreter is a component of a database management system (DBMS) that processes DDL statements to define or modify the structure of a database. It translates DDL commands like `CREATE`, `ALTER`, `DROP`, and `TRUNCATE` into low-level actions that create, modify, or delete database objects such as tables, indexes, and views. The DDL interpreter updates metadata to reflect these changes and enforces data integrity constraints specified in the DDL statements. Its main role is to manage the schema and structure of the database based on user-defined commands.

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

Ans: Data Manipulation Language (DML) statements like `SELECT`, `INSERT`, `UPDATE`, and `DELETE`,

- **Query Parser:** Checks the syntax and structure of DML statements.

- **Query Optimizer:** Analyzes and determines the most efficient way to execute DML statements.

- **Query Executor:** Interacts with the database storage engine to perform data manipulation operations specified by the DML statements.

- **Transaction Manager:** Manages transactions to ensure data integrity and consistency during DML operations.

**8. What is SQL Key Constraints writing an Example of SQL Key Constraints?**

Ans: SQL key constraints are rules or conditions applied to columns in a database table to enforce data integrity and define relationships between tables. There are several types of key constraints in SQL:

1. **Primary Key Constraint:**

- A primary key constraint uniquely identifies each record in a table and ensures that there are no duplicate values in the specified column or columns.

- Example:

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(50),

age INT

);

2. **Unique Key Constraint:**

- A unique key constraint ensures that all values in a specified column or columns are unique (i.e., no duplicates), but unlike a primary key, it allows `NULL` values (except for columns defined as `NOT NULL`).

- Example:

CREATE TABLE Employees (

employee\_id INT UNIQUE,

employee\_name VARCHAR(50),

department\_id INT

);

3. **Foreign Key Constraint:**

- A foreign key constraint establishes a relationship between two tables by enforcing referential integrity. It ensures that values in a column (or columns) of one table match the values in another table's primary key or unique key.

- Example:

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(50),

email VARCHAR(100)

);

**9. What is save Point? How to create a save Point write a Query?**

Ans: A save point in SQL is a point within a transaction where you can set a marker to which you can later roll back if necessary, while still allowing portions of the transaction to be committed. This feature is useful when you want to divide a transaction into smaller segments.

Here's an example of how to create and use a save point within a transaction:

-- Start a transaction

START TRANSACTION;

-- Execute SQL statements within the transaction

INSERT INTO employees (id, name, salary) VALUES (1, 'John', 50000);

INSERT INTO employees (id, name, salary) VALUES (2, 'Jane', 60000);

-- Create a savepoint named 'before\_update'

SAVEPOINT before\_update;

-- Update salary for employee with ID 1

UPDATE employees SET salary = 55000 WHERE id = 1;

-- Check the updated records

SELECT \* FROM employees;

-- Check the records after rolling back

SELECT \* FROM employees;

-- Commit the transaction

COMMIT;

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

Ans: In SQL, a trigger is a special type of stored procedure that automatically executes in response to specific events (like `INSERT`, `UPDATE`, `DELETE`) occurring on a database table. Triggers are used to enforce business rules, validate data, maintain integrity, or automate tasks based on data changes.

To create a trigger:

- Use `CREATE TRIGGER` statement with specified timing (`BEFORE` or `AFTER`) and event (`INSERT`, `UPDATE`, `DELETE`).

- Specify the table on which the trigger will act (`ON table\_name`).

- Define the trigger's actions within `BEGIN` and `END` block.

Example:

CREATE TRIGGER update\_last\_updated

AFTER UPDATE

ON employees

FOR EACH ROW

BEGIN

UPDATE employees

SET last\_updated = CURRENT\_TIMESTAMP

WHERE id = :OLD.id;

END;