Basics of Database

1. What do you understand By Database.

Ans.

Database is a collection of inter-related data and Management System is a set of programs to store and retrieve those data.DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.

2. What is Normalization?

Ans.

Normalization is the process of minimizing redundancy (duplicity) from a relation or set of relations.Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations.

3. What is Difference between DBMS and RDBMS?

Ans.

DBMS (Database Management System) and RDBMS (Relational Database Management System) are both systems that manage databases.

DBMS-

- It can manage any type of data, and the relationships between data may not be well-defined.
- Focuses less on maintaining the integrity of relationships between data.

- Generally provides less flexibility in handling complex relationships between data.
- May or may not support normalization.
- EX. Microsoft Access.

RDBMS-

- It specifically manages data in a relational format, organizing information into tables with predefined relationships between them.
- Enforces the integrity of relationships through the use of primary keys, foreign keys, and other constraints.
- Offers greater flexibility in managing complex relationships, making it more suitable for applications with intricate data dependencies.
- Typically supports normalization to eliminate redundancy and ensure data integrity.
- Ex. MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

4. What is MF Cod Rule of RDBMS Systems?

Ans.

Rule 1: The Information Rule - All information, whether it is user information or metadata, that is stored in a database must be entered as a value in a cell of a table. It is said that everything within the database is organized in a table layout. **Rule 2:** The Guaranteed Access Rule - Each data element is guaranteed to be accessible logically with a combination of the table name, primary key (row value), and attribute name (column value). **Rule 3:** Systematic Treatment of NULL Values - Every Null value in a database must be given a systematic and uniform treatment. **Rule 4:** Active Online Catalog Rule - The database catalog, which contains metadata about the database, must be stored and accessed using the same relational database management system. **Rule 5:** The Comprehensive Data Sublanguage Rule - A crucial component of any efficient database system is its ability to offer an easily understandable data manipulation language (DML) that facilitates defining, querying, and modifying information within the database. **Rule**

6: The View Updating Rule - All views that are theoretically updatable must also be updatable by the system. **Rule 7:** High-level Insert, Update, and Delete - A successful database system must possess the feature of facilitating high-level insertions, updates, and deletions that can grant users the ability to conduct these operations with ease through a single query. **Rule 8:** Physical Data Independence - Application programs and activities should remain unaffected when changes are made to the physical storage structures or methods. **Rule 9:** Logical Data Independence - Application programs and activities should remain unaffected when changes are made to the logical structure of the data, such as adding or modifying tables. **Rule 10:** Integrity Independence - Integrity constraints should be specified separately from application programs and stored in the catalog. They should be automatically enforced by the database system. **Rule 11:** Distribution Independence - The distribution of data across multiple locations should be invisible to users, and the database system should handle the distribution transparently. **Rule 12**: Non-Subversion Rule - If the interface of the system is providing access to low-level records, then the interface must not be able to damage the system and bypass security and integrity constraints.

5. What do you understand By Data Redundancy?

Ans.

Data redundancy occurs when the same piece of data is stored in multiple places within a database. This duplication can lead to inefficiency, increased storage requirements, and a higher risk of inconsistencies or errors in the data.

6. What is DDL Interpreter?

Ans. It interprets the DDL (Data Definition Language) Instructions and stores the record in a data dictionary (in a table containing meta-data).

7. What is DML Compiler in SQL?

Ans. A DML (Data Manipulation Language) compiler in SQL is a component that processes and executes commands related to data manipulation, such as querying, inserting, updating, and deleting records in a database. It translates Data Manipulation Language statements into executable instructions, allowing users to interact with and modify the data stored in the database.

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

Ans. SQL key constraints are rules applied to columns in a relational database table to ensure the integrity and uniqueness of the data. There are different types of key constraints: Primary Key, Unique Key, and Foreign Key.

1.Primary Key Constraint:

- Ensures unique identification of each record in a table.
- No NULL values allowed.
- Example:

```
CREATE TABLE Books (
    ISBN VARCHAR(13) PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(50),
    PublicationYear INT
);
```

2.Unique Key Constraint:

- Ensures that the values in a column (or a combination of columns) are unique.

- Allows NULL values (except in the columns under constraint).
- Example:

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15) UNIQUE,
    Country VARCHAR(50)
);
```

3. Foreign Key Constraint:

- Establishes a link between two tables based on a columns
- Ensures referential integrity by enforcing relationships between tables.
- Example:

```
CREATE TABLE OrderItems (
    OrderItemID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

```
CREATE TABLE Products (
   ProductID INT PRIMARY KEY,
   ProductName VARCHAR(50),
   Price DECIMAL(10, 2)
);
```

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

Ans. savepoint in a relational database is a point within a transaction where you can mark and later roll back to if needed. It allows you to create a named point in the transaction so that you can partially commit the changes and later undo only the modifications made after that savepoint.

In SQL, you can use the `SAVEPOINT` statement to create a savepoint within a transaction. Here's an example:

```
-- Start a transaction
START TRANSACTION;

-- Make some changes
UPDATE Employees SET Salary = Salary + 500 WHERE Department = 'IT';

-- Create a savepoint named 'salary_update'
SAVEPOINT salary_update;

-- Make more changes
```

```
UPDATE Employees SET Bonus = Bonus + 100 WHERE Department = 'Sales';

-- If needed, roll back to the savepoint, undoing only changes after it
ROLLBACK TO SAVEPOINT salary_update;

-- Commit the transaction
COMMIT;
```

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

Ans. 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 typically include actions like INSERT, UPDATE, DELETE, or even a combination of these. Triggers are useful for enforcing business rules, validating data changes, or automating tasks when specific database events occur.

Here's a basic example of how to create a trigger in SQL:

```
-- Creating a simple trigger
CREATE TRIGGER after_employee_insert
AFTER INSERT ON Employees
FOR EACH ROW
BEGIN
    -- Trigger code to be executed after each INSERT on the Employees table
    INSERT INTO AuditLog (Event, TableName, EmployeeID, Timestamp)
    VALUES ('INSERT', 'Employees', NEW.EmployeeID, NOW());
END;
```