MODULE: 5 (Database)
Basics of Database

1. What do you understand By Database

A database is like a super organized digital filing cabinet where all your information is stored neatly. Think of it like a bunch of spreadsheets, each with rows and columns, so you can easily find what you need.

Every database comes with a plan, called a schema, that decides how everything will be arranged. It's like having a blueprint that tells you what kind of info can be stored, any rules that need to be followed, and how everything is connected.

Using a database keeps your data safe and makes sure only the right people can access it. Plus, it's like having a backup buddy – if anything goes wrong and you lose your data, you can usually get it back thanks to special database tools.

So basically, databases are like the secret heroes of modern tech, helping keep our info organized and safe, whether it's your family photos or a big company's records.

2. What is Normalization?

Normalization is like tidying up your data closet to make everything neat and organized. It's a process used in databases to reduce redundancy and dependency by breaking down large tables into smaller, related ones. This makes it easier to manage and ensures data integrity.

Imagine you have a table full of messy data. Normalization helps you split it into smaller tables, each containing specific types of information. This prevents duplicate data and makes updates and searches faster and more efficient.

So, normalization is like the Marie Kondo of databases, making sure everything has its proper place and nothing is cluttering up the space unnecessarily.

3. What is Difference between DBMS and RDBMS?

here are five key differences between DBMS and RDBMS:

Data Organization:

DBMS: Stores data without imposing any structure. It's like a digital storage box where you can put in any type of data.

RDBMS: Organizes data into structured tables with rows and columns, following a relational model. It's like having separate digital folders for different types of information.

Data Relationships:

DBMS: Doesn't enforce relationships between different sets of data.

RDBMS: Maintains relationships between tables through keys, ensuring data integrity and enabling efficient querying.

Normalization:

DBMS: Doesn't typically involve normalization processes.

RDBMS: Utilizes normalization techniques to reduce redundancy and dependency, improving data consistency and efficiency.

Querying:

DBMS: Supports basic data retrieval and manipulation operations.

RDBMS: Provides advanced querying capabilities through SQL (Structured Query Language), allowing for complex data retrieval and analysis.

Scalability and Performance:

DBMS: May lack scalability and performance optimization features.

RDBMS: Often includes features for scalability, indexing, and query optimization, making it suitable for handling large datasets and high-performance applications.

4. What is MF Cod Rule of RDBMS Systems?

MF Codd's 12 rules are like a checklist that defines what a true relational database should be capable of. Think of them as the golden standards for how data should be handled in a database.

Here's a simpler breakdown:

1. **Data Structure**: All data should be organized into tables with rows and columns.

2. **Access Guarantee**: You should be able to access any piece of data easily using specific identifiers.

3. **Handling Missing Data**: There should be a consistent way to deal with missing or unknown information.

4. **Catalog Consistency**: Information about the database's structure should be stored in a way that's consistent with the data itself.

5. **One Language**: You should be able to interact with the database using a single, common language.
6. **View Update**: If you can see it, you should be able to update it.

7. **Bulk Operations**: You should be able to make changes to a whole set of data, not just individual pieces.

8. **Data Storage Independence**: The way data is physically stored shouldn't affect how you use it.

9. **Data Schema Independence**: Changes to how the data is structured shouldn't mess up the applications using it.

10. **Integrity Rules**: Rules about how data should behave should be built into the system, not the applications using it.

11. **Location Flexibility**: The database should be able to work across different physical locations without breaking.

12. **No Sneaky Shortcuts:** If there's a way to bypass the rules, it shouldn't be easy to do.

These rules are like a guide for building databases that are reliable, flexible, and easy to work with. If a database system follows all these rules, it's considered a top-notch relational database management system (RDBMS).

5. What do you understand By Data Redundancy?

Data redundancy is like having duplicate copies of the same information scattered around. Imagine you have a list of your favorite recipes written down in three different notebooks. Each notebook has the same recipes written in them. That's redundancy!

In simpler terms, data redundancy happens when the same piece of information is stored multiple times in a database or across different files or systems. It can lead to confusion, waste of storage space, and inconsistencies if the duplicated data is not properly managed. It's like having extra copies of things you don't really need, cluttering up your digital space.

6. What is DDL Interpreter?

Think of a DDL interpreter like a language translator for databases. Just as you might use a translator to convert words from one language to another, a DDL (Data Definition Language) interpreter helps translate commands that define the structure of a database.

In simpler terms, DDL interpreter is a tool that understands and executes commands used to create, modify, and delete the structure of a database, such as tables, indexes, and constraints. It's like the behind-the-scenes magician that brings your database design ideas to life, turning them into the actual framework of your data storage system.

7. What is DML Compiler in SQL?

DML (Data Manipulation Language) compiler in SQL is like a skilled craftsman who takes your instructions for changing or retrieving data and turns them into actions that the database can understand and execute.

When you want to add, update, delete, or retrieve data from a database using SQL commands, the DML compiler is the one responsible for translating those commands into low-level instructions that the database engine can process. It's like giving a set of blueprints to a builder, and they use their expertise to construct exactly what you need.

So, in essence, the DML compiler is the bridge between your high-level data manipulation commands and the inner workings of the database system, making sure your data changes happen smoothly and accurately.

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

SQL key constraints are rules defined on columns in a table that enforce data integrity and maintain the relationships between tables. They ensure that certain conditions are met regarding the values stored in these columns.

Here's an example of SQL key constraints:

```
CREATE TABLE Employees (
        EmployeeID INT PRIMARY KEY,
        FirstName VARCHAR (50),
        LastName VARCHAR (50),
        DepartmentID INT,
        CONSTRAINT fk_DepartmentID FOREIGN KEY (DepartmentID)
        REFERENCES Departments (DepartmentID)
);
```

In this example:

EmployeeID is defined as the **primary key**, ensuring each employee record has a unique identifier.

DepartmentID is a **foreign key** that references the DepartmentID column in the Departments table. This establishes a relationship between the Employees and Departments tables, ensuring that every DepartmentID value in the Employees table corresponds to a valid DepartmentID in the Departments table.

These key constraints maintain the integrity of the data by enforcing rules such as uniqueness and referential integrity, ensuring consistency and accuracy within the database.

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

A savepoint in SQL is a named marker that represents a point within a transaction to which you can rollback if necessary. It allows you to set points in your transaction where you can partially undo changes without rolling back the entire transaction.

To create a savepoint in SQL, you can use the SAVEPOINT command followed by the name of the savepoint. Here's an example query:

```
SAVEPOINT my_savepoint;
```

In this example, my_savepoint is the name of the savepoint. Once you create the savepoint, you can later rollback to it if needed, using the ROLLBACK TO command followed by the name of the savepoint:

```
ROLLBACK TO my_savepoint;
```

This command will undo changes made after the savepoint was created, but it won't affect changes made before the savepoint or changes made after the savepoint was rolled back to. It's useful for reverting specific parts of a transaction while preserving other changes.

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

In SQL, a trigger is a special type of stored procedure that automatically executes in response to certain events occurring in the database. These events can include actions such as insertions, updates, or deletions of data in a table. Triggers are useful for enforcing data integrity, auditing changes, or performing complex business logic automatically.

To create a trigger in SQL, you use the CREATE TRIGGER statement followed by the trigger name, the event that triggers the execution of the trigger (e.g., AFTER INSERT, BEFORE UPDATE, etc.), and the table or tables on which the trigger should be activated.

Here's a basic example:

```
CREATE TRIGGER log_new_order
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
        INSERT INTO Order_Logs (OrderID, OrderDate, CustomerID, TotalAmount)
        VALUES (NEW.OrderID, NEW.OrderDate, NEW.CustomerID,
    NEW.TotalAmount);
END;
```

In this example:

log_new_order is the name of the trigger.

AFTER INSERT ON Orders specifies that the trigger should execute after an insertion occurs in the Orders table.

FOR EACH ROW indicates that the trigger should execute once for each row affected by the triggering event.

The BEGIN and END keywords enclose the logic or actions to be performed when the trigger is activated.

Inside the trigger logic block, we're inserting a new row into the Order_Logs table whenever a new row is inserted into the Orders table. We're copying the relevant data (OrderID, OrderDate, CustomerID, TotalAmount) from the newly inserted row (NEW) into the Order_Logs table. This way, we maintain a log of all new orders that are added to the system.