Module – 5.1

➢ Basic Of Database

1. What do you understand By Database

- Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc.
- it can be stored in pieces of paper or electronic memory, etc.
- A Database is a structured collection of data that is organized and stored in a way that makes it easy to manage, retrieve, and update.
- A **database** is an organized collection of data, so that it can be easily accessed and managed.
- You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

2. What is Normalization?

- A large database defined as a single relation may result in data duplication.
- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

3. What is Difference between DBMS and RDBMS?

DBMS	RDBMS
DBMS stores data as file.	RDBMS stores data in tabular form.
Data elements need to access individually.	Multiple data elements can be accessed at the same time.
No relationship between data.	Data is stored in the form of tables which are related to each other.
Normalization is not present.	Normalization is present.

DBMS	RDBMS
DBMS does not support distributed database.	RDBMS supports distributed database.
It stores data in either a navigational or hierarchical form.	It uses a tabular structure where the headers are the column names, and the rows contain corresponding values.
It deals with small quantity of data.	It deals with large amount of data.
Data redundancy is common in this model.	Keys and indexes do not allow Data redundancy.
It is used for small organization and deal with small data.	It is used to handle large amount of data.
Not all Codd rules are satisfied.	All 12 Codd rules are satisfied.
Security is less	More security measures provided.
It supports single user.	It supports multiple users.
Data fetching is slower for the large amount of data.	Data fetching is fast because of relational approach.
The data in a DBMS is subject to low security levels with regards to data manipulation.	There exists multiple levels of data security in a RDBMS.
Low software and hardware necessities.	Higher software and hardware necessities.
Examples: XML, Window Registry, Forxpro, dbaseIIIplus etc.	Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

4. What is MF Cod Rule of RDBMS Systems?

- Codd's Rules, proposed by Dr. E.F. Codd, define the requirements that a relational database management system (RDBMS) must meet in order to be considered truly relational.
- Here is a summary of Codd's 12 rules for a relational database system:

I. **Information Rule:**
   - All information in the database is to be represented in one and only one way—namely, as values in a table.

II. **Guaranteed Access Rule:**
   - Each unique piece of data must be accessible by specifying a table name, primary key value, and column name.

III. **Systematic Treatment Of Null Values:**
   - The DBMS must allow each field to remain null (or empty). Null is distinct from an empty string or zero.

IV. **Dynamic Online Catalog Based On the Relational Model:**
   - The structure of the database (metadata) must be stored in tables, accessible using the same query language used for regular data manipulation.

V. **Comprehensive Data Sublanguage Rule:**
   - A relational system must support a language that is complete in terms of relational operations and data definition.

VI. **View Updating Rule:**
   - All views that are theoretically updatable must be updatable by the system.

VII. **High-level Insert, Update, and Delete:**
   - The capability of handling a base relation or a derived relation as a single operand, producing a new relation.

VIII. **Physical Data Independence:**
   - The physical storage structure of the data should not affect the access to the data.

IX. **Logical Data Independence:**
   - Changes in the logical structure (tables, columns, etc.) should not affect the application programs.

X. **Integrity Independence:**
   - Integrity constraints should be stored in the data dictionary and not be a part of the application programs.

XI. **Distribution Independence:**

- user's view of the database should not be affected by the way the data is distributed and stored.

XII. Non-subversion Rule:

- If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the integrity rules of the relational database.

5. What do you understand By Data Redundancy?

- Data redundancy means the occurrence of duplicate copies of similar data.
- It is done intentionally to keep the same piece of data at different places, or it occurs accidentally.
- In DBMS, when the same data is stored in different tables, it causes data redundancy.
- Sometimes, it is done on purpose for recovery or backup of data, faster access of data, or updating data easily.
- Redundant data costs extra money, demands higher storage capacity, and requires extra effort to keep all the files up to date.
- Sometimes, unintentional duplicity of data causes a problem for the database to work properly, or it may become harder for the end user to access data.
- Redundant data unnecessarily occupy space in the database to save identical copies, which leads to space constraints, which is one of the major problems.
- Let us understand redundancy in DBMS properly with the help of an example.

Student_id	Name	Course	Session	Fee	Department
101	Devi	B. Tech	2022	90,000	CS
102	Sona	B. Tech	2022	90,000	CS
103	Varun	B. Tech	2022	90,000	CS
104	Satish	B. Tech	2022	90,000	CS
105	Amisha	B. Tech	2022	90,000	CS

- In the above example, there is a "Student" table that contains data such as "Student_id", "Name", "Course", "Session", "Fee", and "Department".
- As you can see, some data is repeated in the table, which causes redundancy.
- Problems that are caused due to redundancy in the database
- Redundancy in DBMS gives rise to anomalies, and we will study it further.
- In a database management system, the problems that occur while working on data include inserting, deleting, and updating data in the database.

6. What is DDL Interpreter?

- In simple terms, a DDL (Data Definition Language) interpreter is a software component or tool that processes and executes commands related to the structure or definition of a database.
- DDL commands are used to create, modify, or delete database objects such as tables, indexes, and constraints.
- The DDL interpreter takes these commands and performs the necessary actions to implement the specified changes in the database schema.
- It helps in managing the organization and structure of the data within a database by allowing users to define the layout and relationships of the various elements in the database.
- Common DDL commands include "CREATE" (to create a new database object), "ALTER" (to modify the structure of an existing object), and "DROP" (to delete an object).
- The DDL interpreter ensures that these commands are carried out accurately and efficiently, helping users define and maintain the database structure according to their requirements.

7. What is DML Compiler in SQL?

- In SQL (Structured Query Language), there isn't typically something referred to specifically as a "DML Compiler."
- However, there is a concept related to the execution of data manipulation operations called the "query processor" or "query compiler."
- In simple terms, a query processor or compiler in SQL is responsible for translating and optimizing the SQL queries written by users.
- It takes the SQL statements, including Data Manipulation Language (DML) commands like SELECT, INSERT, UPDATE, and DELETE, and converts them into a set of instructions that the database engine can execute to retrieve, modify, or delete data.

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

- In SQL, key constraints are used to enforce the uniqueness and integrity of data within a table.
- There are primarily three types of key constraints: PRIMARY KEY, UNIQUE, and FOREIGN KEY.

i. **PRIMARY KEY Constraint:**

- Ensures that each row in a table is uniquely identified by a specific column or set of columns
- Automatically implies uniqueness and does not allow NULL values.

Example:

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50)
);
```

ii. **UNIQUE Constraint:**

- Ensures that the values in a specified column or set of columns are unique across the table.
- Allows NULL values, but only one NULL value is allowed (as NULL is not considered equal to NULL).

Example:

```
CREATE TABLE Employees (
   EmployeeID INT UNIQUE,
   FirstName VARCHAR(50),
   LastName VARCHAR(50)
);
```

iii. **FOREIGN KEY Constraint:**

- Establishes a link between two tables by referencing a column or set of columns in one table to the primary key of another table.
- Ensures referential integrity.

Example:

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

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Price DECIMAL(10, 2)
);
```

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

- a savepoint is a point in a transaction where you can temporarily save the current progress, and later, if needed, roll back to that specific point without affecting the entire transaction.
- It's like creating a checkpoint within a larger set of database operations.
- To create a savepoint in SQL, you use the SAVEPOINT statement.
- Here's a simple explanation and

  Example:

  Creating a Savepoint:

  ```
  SAVEPOINT your_savepoint_name;
  ```

  In this query:

- **'SAVEPOINT'** is the keyword indicating that you want to create a savepoint.
- **'your_savepoint_name'** is a user-defined name for the savepoint.
- You can choose any name that makes sense to you.

  Example:

```
-- Start of the transaction
BEGIN;

-- Your database operations
UPDATE Employees SET Salary = Salary * 1.1 WHERE Department = 'IT';

-- Creating a savepoint
SAVEPOINT my_update_savepoint;

-- More operations
UPDATE Employees SET Salary = Salary * 1.05 WHERE Department = 'HR';

-- If an issue arises, you can roll back to the savepoint
ROLLBACK TO my_update_savepoint;
```

```
  -- Commit the changes if everything is okay
  COMMIT;
```

- In this example, the savepoint ‘my_update_savepoint’ is created after the first update operation

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

- In SQL, a trigger is a set of instructions that automatically runs (or "triggers") in response to a specific event on a particular table or view.
- These events can include data modifications (INSERT, UPDATE, DELETE) or certain database operations.
- Triggers are useful for enforcing data integrity rules, automating actions, or logging changes.
- **Creating a Trigger :**

```
CREATE TRIGGER trigger_name
ON table_name
[FOR | AFTER | INSTEAD OF] [INSERT | UPDATE | DELETE]
AS
BEGIN
   -- SQL statements to be executed when the trigger is fired
   -- This could include conditions, modifications, or any other SQL operation
END;
```

- **‘trigger_name’ :** Choose a name for your trigger.
- **‘table_name’ :** Specify the table on which the trigger should be activated.
- **‘FOR | AFTER | INSTEAD OF’ :** Define when the trigger should be executed.
  - **‘FOR’ and ‘AFTER’** mean the trigger fires after the specified event.
  - **‘INSTEAD OF’** is used for triggers on views and specifies that the trigger should replace the usual action of the event.
- **‘INSERT | UPDATE | DELETE’ :** Specify the type of event that triggers the action.