Module – 5.1

(Basics 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 an organized collection of data, so that it can be easily accessed and managed.
- It is designed to efficiently store, retrieve, and manage data, making it easy to organize and manipulate information.
- Databases are used in various applications, such as websites, business systems, and software, to store and retrieve data in a systematic and efficient manner.

2. What is Normalization?

- 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?

No.	DBMS	RDBMS
1)	DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	**Normalization is not** present in DBMS.	**Normalization is** present in RDBMS.
4)	DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.

5)	DBMS uses file system to store data, so there will be **no relation between the tables**.	in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
8)	DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.	RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
9)	Examples of DBMS are file systems, **xml** etc.	Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc.

4. What is MF Cod Rule of RDBMS Systems?

- Codd's Rules, also known as Codd's Twelve Rules, were proposed by Dr. E.F. Codd, the inventor of the relational database model.
- These rules were designed to define what characteristics a system must have to be considered a true relational database management system (RDBMS).
- Here is a summary of Codd's Rules:

1) **Information Rule:**
   - All information in the database is to be stored in one and only one place, the relational database.
2) **Guaranteed Access Rule:**
   - Each unique piece of data (atomic value) is accessible by specifying the table name, primary key value, and column name.
3) **Systematic Treatment of Null Values:**
   - The DBMS must allow each field to remain null (or empty).
4) **Dynamic Online Catalog Based on the Relational Model:**
   - The database schema, including metadata such as table definitions and constraints, is stored in a catalog that is accessible to users.
5) **Comprehensive Data Sublanguage Rule:**
   - The DBMS must support a data sublanguage that is comprehensive, meaning it can express any complex query or operation.
6) **View Updating Rule:**
   - All views that are theoretically updatable must be updatable by the system.

7) **High-Lavel Insert, Update, and Delete:**
   - The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update, and deletion of data.
8) **Physical Data Independence:**
   - Changes in the physical storage structure or devices should not affect the access to the stored data.
9) **Logical Data Independence:**
   - Changes to the logical schema (table structures, constraints) should not affect existing applications.
10) **Integrity Independence:**
   - Integrity constraints must be definable separately from application programs and stored in the catalog.
11) **Distribution Independence:**
   - A user's perception of the data should not be altered by the way the data is physically distributed or stored.
12) **Nonsubversion Rule:**
   - If a relational system has a low-level language, that low-level language cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher-level relational language.

5. What do you understand By Data Redundancy?

- **Redundancy** means having multiple copies of the same data in the database. This problem arises when a database is not normalized.
- Suppose a table of student details attributes is: student ID, student name, college name, college rank, and course opted.

Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	B.Tech	1
101	Ankit	7900734858	GEU	B.Tech	1
102	Ayush	7300936759	GEU	B.Tech	1
103	Ravi	7300901556	GEU	B.Tech	1

- It can be observed that values of attribute college name, college rank, and course are being repeated which can lead to problems.

- **Enhanced Query Performance:** By eliminating the need for intricate joins, redundancy helps expedite data retrieval.
- **Offline Access:** In offline circumstances, redundant copies allow data access even in the absence of continuous connectivity.
- **Increased Availability:** Redundancy helps to increase fault tolerance, which makes data accessible even in the event of server failures.

6. What is DDL Interpreter?

- A DDL (Data Definition Language) Interpreter is a tool or component in a database management system (DBMS) that handles the execution and processing of Data Definition Language statements.
- DDL statements are used to define and manage the structure and organization of a database, including creating, modifying, and deleting database objects such as tables, indexes, and constraints.
- A DDL Interpreter is responsible for understanding and executing commands that define the structure of a database, ensuring that the database schema reflects the desired organization and relationships between different data elements.
- It plays a crucial role in managing the metadata of a database, which describes how data is stored, accessed, and related within the system.

7. What is DML Compiler in SQL?

- DML (Data Manipulation Language) statements are executed directly by the database engine or SQL interpreter, without the need for a separate compiler.
- DML statements in SQL are used to manipulate data stored in the database, such as retrieving, inserting, updating, or deleting records from tables.
- These statements include SELECT, INSERT, UPDATE, and DELETE.
- DML statements in SQL are typically executed directly by the SQL engine, and there is no separate DML Compiler in the traditional sense.

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

- In SQL, key constraints are rules or conditions applied to columns in a table to enforce the relationships between the tables and maintain data integrity.
- There are three primary types of key constraints: Primary Key, Foreign Key, and Unique Key.

1) **Primary Key :**
    - A Primary Key uniquely identifies each record in a table.
    - It must contain unique values, and it cannot have NULL values.
    - Every table can have only one Primary Key.
    - EXAMPLE-

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   StudentName VARCHAR(50),
   Age INT
);
```

2) **Foreign Key :**
   - A Foreign Key establishes a link between two tables by referencing the Primary Key of another table.
   - It ensures referential integrity between the linked tables.
   - The values in the Foreign Key must match the values in the referenced Primary Key.
   - EXANPLE-

```
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(50),
   Price DECIMAL(10, 2)
);
```

3) **Unique Key :**
   - A Unique Key ensures that all values in a column are distinct.
   - Unlike the Primary Key, a table can have multiple Unique Keys.
   - NULL values are allowed, but only one NULL value is permitted.
   - EXAMPLE-

```
CREATE TABLE Employees (
   EmployeeID INT UNIQUE,
   EmployeeName VARCHAR(50),
   DepartmentID INT
);
```

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

- In a database, a savepoint is a point within a transaction where you can mark your progress and later roll back to that point if needed.

- It allows you to create a kind of "checkpoint" during a transaction, so you can undo part of your changes without rolling back the entire transaction.
- A savepoint is like a bookmark in your transaction.
- It allows you to roll back to that specific point if something goes wrong.
- Useful when you want to undo changes up to a certain stage without affecting the entire transaction.
- EXAMPLE-

```
-- Start a transaction
START TRANSACTION;

-- SQL statements...

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

-- More SQL statements...

-- If something goes wrong, you can roll back to the savepoint
ROLLBACK TO my_savepoint;

-- Additional SQL statements...

-- If everything is successful, commit the transaction
COMMIT;
```

- In this example:
  - 'START TRANSACTION' begins a new transaction.
  - 'SAVEPOINT my_savepoint' creates a savepoint named 'my_savepoint'.
  - 'ROLLBACE TO my_savepoint' rolls back the transaction to the savepoint if needed.
  - 'COMMIT' finalizes the transaction if everything is successful.

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

- In SQL, a trigger is a set of instructions or code that automatically runs in response to a specific event on a particular table or view.
- Triggers are used to enforce business rules, perform validations, or automate certain actions when data in the database changes.
- A trigger is like a stored procedure associated with a specific table or view.
- It is automatically executed (or "triggered") when a specified event occurs, such as an insert, update, or delete operation on the associated table.

- Triggers are useful for enforcing data integrity, performing validation checks, or automating actions based on changes in the database.
- EXAMPLE-

```
-- Create a table
CREATE TABLE ExampleTable (
    ID INT PRIMARY KEY,
    Data VARCHAR(50),
    CreatedAt TIMESTAMP
);

-- Create a trigger
CREATE TRIGGER UpdateTimestamp
BEFORE INSERT ON ExampleTable
FOR EACH ROW
SET NEW.CreatedAt = CURRENT_TIMESTAMP;
```