**Module 4 database**

**Basic of database**

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

🡪A database management system (or DBMS) is essentially nothing more than a computerized data-keeping system

🡪Database Management System (DBMS) is software used to identify, manage, and create a database that provides administered access to the data..

**2. What is Normalization?**

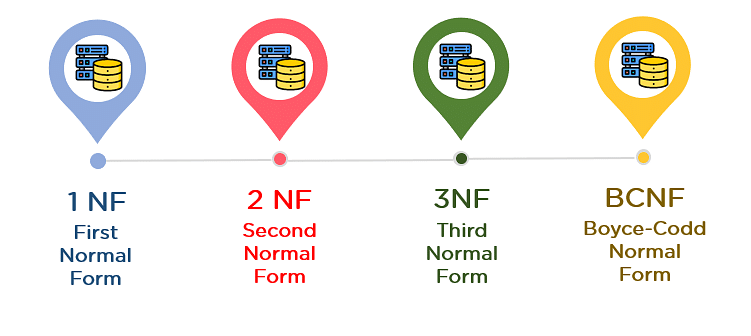
- Normalization is the process of organizing the data in the database

🡪. 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.



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

**1 Dbms:-** DBMS (Database Management System): A DBMS is a software system that manages and organizes data in a way that makes it easy to store, retrieve, and update information.

🡪 It provides an interface between the user and the database**.**

🡪Data can be stored in any format, such as flat files or hierarchical, network, or object-oriented structures.

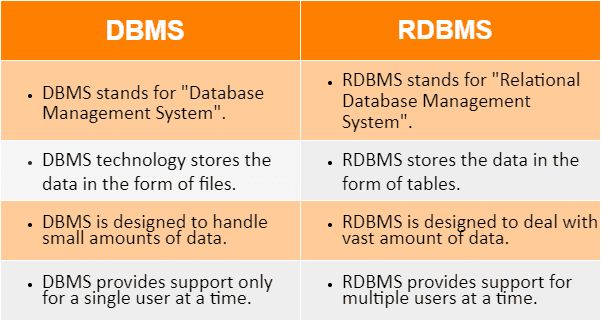
🡪Does not support relationships between tables, so the data is often isolated

1. **RDBMS?**

🡪 **RDBMS (Relational Database Management System):** An RDBMS is a type of DBMS that stores data in a structured format using tables, rows, and columns (also known as relations). It uses the relational model to organize and manage data.

🡪 Data is stored in tables with a predefined schema. These tables consist of rows and columns, where each row represents a record and each column represents an attribute of the record.

🡪 Supports relationships between tables using foreign keys. These relationships allow tables to be linked in meaningful ways, facilitating more complex queries and data manipulation.

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

🡪 The MF (Minimal Functional) Cod Rule is a concept in Relational Database Management Systems (RDBMS) that refers to a set of rules that help in defining the functional dependencies between attributes (columns) in a relational database.

🡪 While the exact term "MF Cod Rule" is not a standard term in relational database theory, I believe you're referring to the Codds' Rules, which were proposed by Edgar F. Codd, the creator of the relational model for databases.

**Key Codds' Rules for RDBMS (Often Known as Codd's Rules):**

1. **Information Rule:** All information in the database is represented explicitly in tables (relations), and every piece of data is stored as a value in a cell in a table. This ensures that the data is logically organized and can be retrieved via queries.
2. **Guaranteed Access Rule:** Every data element is accessible by specifying the name of the table, the name of the column (attribute), and the primary key value of the row. This ensures that the data is accessible through unique identifiers.
3. **Systematic Treatment of Null Values:** The database must allow null values, and these should be treated consistently. Nulls represent unknown or missing data, and they must not be confused with empty strings or zeros.
4. **Dynamic On-Line Catalog Based on the Relational Model:** The database system must provide a catalog (or data dictionary) that contains information about the database schema, data, and relationships. The catalog should be accessible through the same relational model.
5. **Comprehensive Data Sublanguage Rule:** The database must support a comprehensive data manipulation language that includes operations like querying, inserting, updating, and deleting data. SQL is typically the language used for this purpose in RDBMS systems.
6. **View Updating Rule:** All views (virtual tables created from one or more tables) must be updatable, meaning you can modify the underlying data through views (though there are some restrictions based on the view's complexity).
7. **High-Level Insert, Update, and Delete:** The system must allow users to insert, update, or delete data in a single operation or query, and the database must handle the underlying complexities efficiently.
8. **Physical Data Independence:** The application should not be affected by changes in the physical storage of data (e.g., changes in indexing or storage structures). This ensures that users can continue to work without worrying about the physical layer.
9. **Logical Data Independence:** The system should support changes in the logical schema without affecting the application or user interface, making it flexible for modifications in the database schema.
10. **Integrity Independence:** The database system must support integrity constraints (e.g., primary keys, foreign keys, and check constraints) and enforce these constraints to maintain the accuracy and consistency of data.
11. **Distribution Independence:** The database should function correctly regardless of where the data is stored or how it is distributed across machines. Users should not need to worry about the distribution of data across different locations.
12. **Non-Subversion Rule:** If the system provides low-level access to the data, it must not allow users to bypass the relational system’s integrity rules or constraints. This ensures that the relational model's rules are upheld at all times.

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

🡪- Data redundancy refers to the practice of keeping data in two or more places within a database or data storage system. Data redundancy ensures an organization can provide continued operations or services in the event something happens to its data –

🡪 for example, in the case of data corruption or Data Loss. The concept applies to areas such as databases, computer memory and file storage systems.

**6. What is DDL Interpreter**?

A **DDL (Data Definition Language) Interpreter** is a component or part of a database management system (DBMS) that is responsible for interpreting and executing commands written in DDL.

🡪DDL is a subset of SQL (Structured Query Language) used to define and modify database structures, such as tables, schemas, and indexes.

🡪DDL statements only modify the database’s schema; they have no direct effect on the data within the database. • DDL declarations are irreversible and difficult to undo.

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

**🡪**A DML (Data Manipulation Language) Compiler in SQL is a component of a database management system (DBMS) that interprets and processes DML statements.

🡪 DML is used for managing data within the schema objects like tables and views. The DML Compiler is responsible for converting these SQL commands into a form that can be executed by the DBMS

**Common DML Commands:**

* **SELECT**: Retrieves data from one or more tables.
* **INSERT**: Adds new records into a table.
* **UPDATE**: Modifies existing records in a table.
* **DELETE**: Removes records from a table.

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

SQL Key Constraints are rules applied to columns in a database table to enforce data integrity and uniqueness. They ensure that data entered into a database adheres to specific rules and maintains the relational model's integrity.

🡪 There are several types of key constraints in SQL, such as Primary Key, Foreign Key, Unique Key, and Composite Key.

Example :- -- Create 'departments' table with a Primary Key

CREATE TABLE departments ( department\_id INT PRIMARY KEY, -- Primary Key: Uniquely identifies each

department department\_name VARCHAR(50) UNIQUE -- Unique Key: Ensures each department name is unique );

-- Create 'employees' table with a Primary Key and Foreign Key

CREATE TABLE employees ( employee\_id INT PRIMARY KEY, -- Primary Key: Uniquely identifies each employee

first\_name VARCHAR(50),

last\_name VARCHAR(50),

department\_id INT, -- Foreign Key: Links to the departments table CONSTRAINT fk\_department FOREIGN KEY (department\_id) REFERENCES departments(department\_id) -- Foreign Key Constraint ); **9. What is save Point? How to create a save Point write a Query?**

A Savepoint is a feature in SQL that allows you to set a point within a transaction to which you can later roll back if needed, without affecting the entire transaction.

🡪It is useful when you want to commit only parts of a transaction or revert certain changes while keeping others intact.

• Transaction: A series of SQL statements executed as a single unit of work.

. • Savepoint: A marker within a transaction that allows you to roll back part of the transaction instead of rolling back the entire transaction.

**How to Create a SAVEPOINT:**

You can create a savepoint using the SAVEPOINT statement followed by a name for the savepoint.

**How to Create a SAVEPOINT:**

You can create a savepoint using the SAVEPOINT statement followed by a name for the savepoint.

Syntax: 1. Create Savepoint:

SAVEPOINT savepoint\_name;

1. Rollback to Savepoint :-

ROLLBACK TO SAVEPOINT savepoint\_name;

. Release Savepoint (Optional): RELEASE SAVEPOINT savepoint\_name;

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

**🡪**A trigger in SQL is a database object that automatically executes a predefined action (such as inserting, updating, or deleting data) in response to certain events on a table or view.

🡪Triggers are often used to enforce business rules, automate actions, or ensure data integrity**.**

**Types of Triggers:**

**1. Before Trigger:** Executes before an operation (e.g., before an INSERT or UPDATE is made).

2. After Trigger: Executes after an operation (e.g., after a DELETE or UPDATE is completed).

**3. Instead of Trigger:** Used on views to override the behavior of INSERT, UPDATE, or DELETE**.**

-- Create an audit\_log table for storing log information

CREATE TABLE audit\_log (

log\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_id INT, action VARCHAR(50),

timestamp DATETIME

);

-- Create a trigger to log INSERT operations on the employees table CREATE TRIGGER after\_employee\_insert

AFTER INSERT ON employees

FOR EACH ROW

BEGIN –

Insert into audit\_log when a new employee is added

INSERT INTO audit\_log (employee\_id, action, timestamp)

VALUES (NEW.employee\_id, 'INSERT', NOW());

END;