TOPS Technology

Module :- 4 Database

Basic of Database

1. What do you understand By Database ?

Answer :- 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?

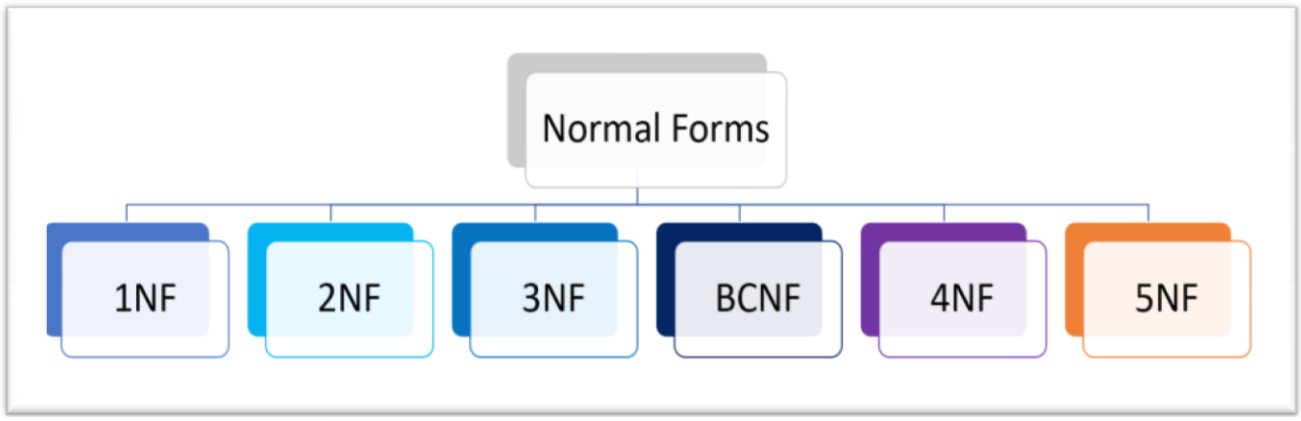
Answer :-

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?

Answer :-

The main difference between a database management system (DBMS) and a relational database management system (RDBMS) is the way data is stored.

Data storage: DBMS stores data in files, while RDBMS stores data in tables.

Data relationships: DBMS data has no relationship or link, while RDBMS uses foreign keys to create relationships between tables.

Data normalization: DBMS doesn't support normalization, but RDBMS does.

Data security: DBMS has low security, while RDBMS is more secure.

Data speed: DBMS is slower than RDBMS.

Data handling: DBMS handles data in navigational or hierarchical formats, while RDBMS uses tables and primary identifiers.

Number of users: DBMS supports single users, while RDBMS supports multiple users.

4. What is MF Cod Rule of RDBMS Systems?

Answer :-

Cod's Rules in DBMS

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.

5. What do you understand By Data Redundancy?

Answer :- 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?

Answer :-

- DDL is used to specify a database's structure, which includes its tables, views, indexes, and constraints.
- DDL commands come in the following types: CREATE, ALTER, DROP, RENAME, and TRUNCATE.
- 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?

Answer :-

A **DML Compiler** in SQL (Data Manipulation Language Compiler) is a component of a database management system (DBMS) that translates high-level DML statements into low-level machine-executable code or intermediate instructions that can be executed by the database engine. DML statements are used to manipulate data in a database, such as inserting, updating, deleting, and retrieving data.

- **`INSERT`**: Adds new rows of data into a table (e.g., `INSERT INTO table_name VALUES (...)`).
- **`UPDATE`**: Modifies existing rows of data in a table (e.g., `UPDATE table_name SET column_name = ... WHERE condition`).
- **`DELETE`**: Removes rows of data from a table (e.g., `DELETE FROM table_name WHERE condition`).
- **`SELECT`**: Retrieves data from the database (e.g., `SELECT column_name FROM table_name WHERE condition`).

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

Answer :-

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?

Answer :-

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

Syntax:

1. **Create Savepoint**:

   SAVEPOINT savepoint_name;

2. **Rollback to Savepoint** :-

   ROLLBACK TO SAVEPOINT savepoint_name;

3. **Release Savepoint** (Optional):

   RELEASE SAVEPOINT savepoint_name;

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

Answer :- 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;
```

Submitted By :- Ansari Amanhusain K.