MODULE: 5 (Database)

1.What do you understand By Database

Ans:- A database is an organized collection of data stored in a computer system and usually controlled by a database management system (DBMS).The data in common databases is modeled in tables, making querying and processing efficient. Structured query language (SQL) is commonly used for data querying and writing. A database is an essential part of our life.

2.What is Normalization?

Ans:- Normalization is a process in database design that aims to eliminate data redundancy and improve data integrity by organizing data into separate related tables. The primary goal of normalization is to minimize the chances of data anomalies, such as insertion, update, and deletion anomalies, and to make the database structure more efficient, maintainable, and scalable. It involves breaking down large, complex tables into smaller, related tables while establishing relationships between them. The process is typically applied in the context of relational databases, where data is organized into tables.

3.What is Difference between DBMS and RDBMS?

Ans:-

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 table which are related to each other.
Normalization is not present.	Normalization is present.
DBMS does not support distributed database.	RDBMS supports distributed database.
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.
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.
Low software and hardware necessities.	Higher software and hardware necessities.
Examples: XML, Window Registry, Forxpro , SQL dbaselllplus etc.	Examples: MySQL, PostgreSQL, Server, Oracle, Microsoft Access etc.

4.What is MF Cod Rule of RDBMS Systems?

Ans:- These rules are made to ensure data integrity, consistency, and usability in a RDBMS. The MF Cod Rule is not a commonly known term in the context of RDBMS systems. However, if you meant Codd's Rule, then it is a set of 12 rules that signify the characteristics and requirements of an RDBMS .

Here are the 12 rules:

(I)The Information Rule

(II)The Guaranteed Access Rule

(III)Systematic Treatment of NULL Values

(IV)Active Online Catalog Rule

(V)The Comprehensive Data Sublanguage Rule

(VI)The View Updating Rule

(VII)High-level Insert, Update, and Delete

(VIII)Physical Data Independence

(IX)Logical Data Independence

(X)Integrity Independence

(XI)Distribution Independence

(XII)Non-Subversion Rule

5.What do you understand By Data Redundancy?

Ans:- Data redundancy refers to the situation in which the same data is stored in multiple places within a database or across multiple databases. It occurs when identical or similar pieces of data are duplicated or repeated, rather than being stored in a single, centralized location. Data redundancy can lead to several problems and challenges in database management.

6.What is DDL Interpreter?

Ans:- DDL Interpreter is a component of the Query Processor in a Database Management System (DBMS). It processes Data Definition

Language (DDL) statements into a set of tables containing metadata, which is stored in the data dictionary.

DDL Interpreter is responsible for building and modifying the structure of tables and other objects in the database.

Here are some steps to use DDL Interpreter:

(i)Open the DBMS software.

(ii)Create a new database or select an existing one.

(iii)Write DDL statements to create, modify, or delete tables or other
(iv)objects in the database.

(v)Submit the DDL statements to the DBMS.

(vi)The DDL Interpreter will process the statements and create or modify tables and other objects in the database

7.What is DML Compiler in SQL?

Ans:- In SQL, DML stands for "Data Manipulation Language," and it is a subset of SQL that is used to manipulate or operate on data stored in a database. DML includes commands for querying and modifying data within database tables. Some common DML commands include:

(i)SELECT: Used to retrieve data from one or more tables.

(ii)INSERT: Used to add new records (rows) to a table.

(iii)UPDATE: Used to modify existing records in a table.

(iv)DELETE: Used to remove records from a table.

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

Ans:- In SQL, key constraints are used to enforce the uniqueness and integrity of data in a table. There are several types of key constraints, including primary keys, unique keys, and foreign keys.

(i)Primary Key Constraint:

-A primary key is a column or a set of columns that uniquely identifies each row in a table.

-It enforces the uniqueness of values in the specified column(s) and ensures that they are not NULL.

-Each table can have only one primary key.

Example of defining a primary key in SQL:

sql

Copy code

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

(ii)Unique Key Constraint:

-A unique key is similar to a primary key but allows for one NULL value in the column(s).

-It enforces the uniqueness of values but doesn't require that values be non-null.

-Each table can have multiple unique keys.

Example of defining a unique key in SQL:

sql

Copy code

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

(iii)Foreign Key Constraint:

-A foreign key is a column or a set of columns that establishes a link between data in two tables.

-It ensures that values in the foreign key column(s) match values in the corresponding primary key column(s) in another table.

-It helps maintain referential integrity between related tables.

Example of defining a foreign key in SQL:

sql

Copy code

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

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

Ans:- A savepoint in a database management system, including SQL databases, is a point within a transaction to which you can later roll back. It allows you to create a point of reference during a transaction and return to that point if needed, effectively undoing changes made in the transaction up to that savepoint.

To create a savepoint in SQL, you use the SAVEPOINT statement. Here's the general syntax:

sql

Copy code

```
SAVEPOINT savepoint_name;
```

Here's an example of how to create a savepoint and then use it to roll back to that point within a SQL transaction:

sql

Copy code

```
-- Start a transaction

BEGIN;


-- Insert some data

INSERT INTO Employees (EmployeeID, EmployeeName) VALUES (1,
'John');

INSERT INTO Employees (EmployeeID, EmployeeName) VALUES (2,
'Jane');


-- Create a savepoint

SAVEPOINT my_savepoint;


-- Continue with more operations

INSERT INTO Employees (EmployeeID, EmployeeName) VALUES (3,
'Bob');

DELETE FROM Employees WHERE EmployeeID = 2;


-- Roll back to the savepoint

ROLLBACK TO my_savepoint;


-- The changes made after the savepoint are undone, but the data
before the savepoint remains intact
```

-- Commit the transaction

COMMIT;

In this example:

(i)We start a transaction using 'BEGIN'.

(ii)Insert some data into the 'Employees' table.

(iii)Create a savepoint named 'my_savepoint' using 'SAVEPOINT'.

(iv)Continue with additional operations, including inserting more data and deleting a record.

(v)When we use 'ROLLBACK TO my_savepoint', the changes made after the 'my_savepoint' are rolled back, but the data before the savepoint remains intact.

(vi)Finally, we commit the transaction using 'COMMIT'.

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

Ans:-In SQL, a trigger is a database object that defines a set of actions to be performed automatically when a certain event occurs within a database table. These events typically involve data manipulation operations such as INSERT, UPDATE, DELETE, or other changes to the table. Triggers are often used to enforce data integrity rules, perform logging, or automate other tasks in response to changes in the database.

Here's how you can create a trigger in SQL:

Syntax for Creating a Trigger:

sql

Copy code

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
[FOR EACH ROW] -- This is specific to some database systems
BEGIN
   -- Trigger actions or logic go here
END;
```

(i)'trigger_name': A unique name for the trigger within the database.

(ii)'BEFORE' or 'AFTER': Specifies whether the trigger should execute before or after the triggering event.

(iii)'INSERT', 'UPDATE' or 'DELETE': Specifies the type of event that triggers the execution of the trigger.

(iv)'ON table_name': Specifies the table on which the trigger should be applied.

(v)'[FOR EACH ROW]': This part is database-specific and indicates that the trigger is a row-level trigger, which means it will execute for each affected row when multiple rows are modified in a single operation.

(vi)'BEGIN' and 'END': Enclose the logic or actions to be performed when the trigger is invoked.

TASK

1. Create Table Name : Student and Exam

Query:- CREATE TABLE student(Rollno INT PRIMARY KEY, NAME VARCHAR(10), Branch VARCHAR(25));

INSERT INTO `student`(`Rollno`, `NAME`, `Branch`)
VALUES(1,"Jay","Computer Science"),(2,"Suhani","Electronic and Com"),
(3,"Kriti","Electronic and Com");

CREATE TABLE Exam(Rollno INT, S_code TEXT, Marks INT, P_code CHARACTER, FOREIGN KEY (Rollno) REFERENCES student(Rollno));

INSERT INTO `exam`(`Rollno`, `S_code`, `Marks`, `P_code`)

VALUES(1, "CS11", 50, "CS"),(1,"CS12",60,"CS"),(2,"EC101",66,"EC"),

(2,"EC102",70,"EC"),(3,"EC101",45,"EC"),(3,"EC102",50,"EC");

2. Create table given below.

Query:- CREATE TABLE Emp(FirstName varchar(10), LastName varchar(10), Address text, City varchar(10), Age int);

INSERT INTO `emp`(`FirstName`, `LastName`, `Address`, `City`, `Age`)
VALUES("Mickey","Mouse","123 Fantasy Way","Anaheim",73),

("Bat","Man","321 Cavern Ave","Gotham",54),
("Wonder","Woman","987 Truth Way","Paradise",39),
("Donald","Duck","555 Quack Street","Mallard",65),
("Bugs","Bunny","567 Carrot Street","Rascal",58),
("Wiley","Coyote","999 Acme Way","Canyon",61),
("Cat","Woman","234 Purrfect Street","Hairball",32),
("Tweety","Bird","543","Itotltaw",28);

3. Create table given below: Employee and Incentive.

Query:- (a)SELECT First_name FROM employee WHERE First_name = 'Tom';

(b)SELECT First_name,Salary,Joining_date FROM employee;

(c)SELECT * FROM employee ORDER BY First_name ASC; AND SELECT * FROM employee ORDER BY Salary DESC;

(d)SELECT * FROM employee WHERE First_name LIKE 'j%';

(e)SELECT Department, MAX(Salary) AS MaxSalary FROM Employee GROUP BY Department ORDER BY MaxSalary ASC;

(f)SELECT First_name, incentive_amount FROM incentive,employee HAVING COUNT(incentive_amount) > 3000;

4. Create table given below: Salesperson and Customer

Query:-

(a)SELECT RATING FROM customer WHERE RATING > 100;

(b)SELECT SNAME,CITY from salesperson WHERE city="london" AND COMM > 0.12;

(c)SELECT * FROM salesperson WHERE CITY="barcelona" OR city ="london";

(d)SELECT * FROM salesperson WHERE COMM BETWEEN .10 AND .12;

(e)SELECT * FROM `customer` WHERE CITY = "roe" and RATING<=100;