Software Engineering Assignment

Module : 5 (Database)

❖ What do you understand about databases ?

A database is an organised collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modelling approaches.

❖ What is normalisation ?

Normalisation is the process of organising the data in the database. Which used to minimise the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.

Normalisation divides the larger table into smaller and links them using relationship. The normal form is used to reduce redundancy from the database table.

In database management system (DBMS), normal forms are a series of guidelines that help to ensure that the design of a database is efficient,organised, and free from data anomalies.

❖ What is the difference between DBMS and RDBMS ?

DBMS	RDBMS
• DBMS (Database Management System)	• RDBMS (Relational Database Management System)
• Data stored in file format.	• Data stored in table format.
• Individual access of data element.	• Multiple data elements are accessible together

• No connection between data	• Data in the form of table are linked together.
• There is only low security while handling data	• It features multiple layers of security while handling data.
• It support single user.	• It supports multiple users
• The software and hardware requirement are low.	• The software and hardware requirement are higher.
• XML, Microsoft Access.	• Oracle, SQL server,...

❖ What is the MF cod rule of RDBMS System ?

Codd’s rule in DBMS also known as Codd's 12 rules/commandments is a set of thirteen rules (numbered 0 to 12) that define a database to be a correct Relational Database Management System (RDBMS). If a database follows Codd's 12 rules, it is called a True relational database management system.

❖ What do you understand about data redundancy ?

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.

Maybe in the database 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 occupies space in the database to save identical copies, which leads to space constraints, which is one of the major problems.

❖ What is a DDL interpreter ?

DDL Interpreter DDL expands to Data Definition Language. DDL Interpreter as the name suggests interprets the DDL statements such as schema definition statements like create, delete, etc. The result of this interpretation is a set of a table that contains the meta-data which is stored in the data dictionary.

❖ What is a DML Compiler in SQL ?

A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database. A DML is typically a sublanguage of a larger database language like SQL, with the DML containing some of the language's operators.

❖ What is SQL Key Constraints writing an Example of SQL Key Constraints ?

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. It can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

➔ The following constraints are commonly used in SQL:

- NOT NULL :- Ensures that a column cannot have a NULL value
- UNIQUE :- Ensures that all values in a column are different
- PRIMARY KEY :- A combination of NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY :- Prevents actions that would destroy links between tables
- CHECK :- Ensures that the values in a column satisfies a specific condition
- DEFAULT :- Sets a default value for a column if no value is specified
- CREATE INDEX :- Used to create and retrieve data from the database very quickly

❖ What is save Point? How to create a save Point write a Query ?

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. This command serves only in the creation of a SAVEPOINT among all the transactional statements. The syntax for a SAVEPOINT command is as shown below.

➔ **SAVEPOINT SAVEPOINT_NAME;**

❖ What is Trigger and how to create a Trigger in SQL ?

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

Task

1. **Create Table Name : Student and Exam**

   ➢ CREATE TABLE student(rollno int PRIMARY KEY AUTO_INCREMENT, name varchar(40), branch varchar(40));

   ➢ CREATE TABLE exam(rollno int, s_code varchar(30), marks int, p_code varchar(30), FOREIGN KEY (rollno) REFERENCES student(rollno));

   ➢ INSERT INTO student(name, branch) VALUES("Jay","Computer Science"), ("Suhani","Electronic and Com."), ("Kirti","Electronic and Com.");

   ➢ 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"), (2,"EC102",50,"EC");

2. Create table given below

➢ CREATE TABLE data(First_Name varchar(20), Last_Name varchar(20), Address varchar(50), City varchar(20), Age int);

➢ INSERT INTO data(First_Name, Last_name, Address, City, Age) values
(“Mickey”,”Mouse”,”123 Fantasy Way”,”Anaheim”,73),
(“Bat”,”Man”,”321 Cavern Ave”,”Gotham”,54),
(“Wonder”,”Women”,”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 Fantasy Street”,”Itotltaw”,28);

3. Create table given below: Employee and Incentive.

➔ **Table name : Employee**

➢ CREATE TABLE employee(employee_id int PRIMARY KEY AUTO_ INCREMENT, first_name varchar(50), last_name varchar(50), salary int, joining_date timestamp, department varchar(50));

➢ INSERT INTO employee(first_name,last_name,salary,joining_date,department) VALUES("john","abraham",100000,now(),"banking"),
("michael","clarke",800000,now(),"insurance"),
("roy","thomas",700000,now(),"banking"),
("tom","jose",600000,now(),"insurance"),
("jerry","pinto",650000,now(),"service"),...;

➔ **Table name : Incentive**

➢ CREATE TABLE incentive(employee_ref_id int, incentive_date datetime, incentive_amount int, FOREIGN KEY (employee_ref_id) REFERENCES employee(employee_id));

➢ INSERT INTO incentive(employee_ref_id,incentive_date,incentive_amount) VALUES(1,now(),5000),
(2,now(),3000),
(3,now(),4000),
(4,now(),4500),...;

a. **Get First_Name from employee table using Tom name “Employee Name”.**
   - ➢ SELECT * FROM employee WHERE first_name = 'tom';

b. **Get FIRST_NAME, Joining Date, and Salary from employee table.**
   - ➢ SELECT first_name, joining_date, salary FROM employee;

c. **Get all employee details from the employee table order by First_Name Ascending and Salary descending?**
   - ➢ SELECT * FROM employee ORDER BY first_name ASC;
   - ➢ SELECT * FROM employee ORDER BY salary DESC;

d. **Get employee details from the employee table whose first name contains ‘J’.**
   - ➢ SELECT * FROM employee WHERE first-name LIKE ‘J%’;

e. **Get department wise maximum salary from employee table order by salary ascending?**
   - ➢ SELECT department FROM employee ORDER BY salary ASC;

f. **Select first_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000**
   - ➢ SELECT employee.first_name, incentive.incentive_amount FROM employee JOIN incentive ON employee.employee_id = incentive .employee_ref_id WHERE incentive.incentive_amount > 3000;

g. **Create After Insert trigger on Employee table which insert records in view table**
   - ➢ CREATE TABLE view_table(id int, name varchar(50), date_time timestamp, record varchar(50));

     DROP TRIGGER IF EXISTS `insert_view_table`;

     CREATE DEFINER=`root`@`localhost` TRIGGER `insert_view_table` AFTER INSERT ON `employee` FOR EACH ROW BEGIN INSERT INTO view_table(id, name, record) VALUES(new.employee_id, new.first_name, "record inserted");

     END

4. **Create table given below: Salesperson and Customer**

   - CREATE TABLE Salesperson(SNo int PRIMARY KEY AUTO_INCREMENT, SNAME varchar(50), CITY varchar(50), COMM float);

   - CREATE TABLE Customer(CNM int PRIMARY KEY AUTO_INCREMENT, CNAME varchar(50), CITY varchar(50), RATING int, SNo int, FOREIGN KEY (SNo) REFERENCES Salesperson(SNo));

   a. **All orders for more than $1000.**
      - This valuable data is not in the table.

   b. **Names and cities of all salespeople in London with commission above 0.12**
      - SELECT SNAME, CITY, COMM FROM Salesperson WHERE CITY = ”London” AND COMM > 0.12;

   c. **All salespeople either in Barcelona or in London**
      - SELECT * FROM Salesperson WHERE CITY = ”Barcelona” OR CITY = ”London”;

   d. **All salespeople with commission between 0.10 and 0.12. (Boundary values should be excluded).**
      - SELECT * FROM Salesperson WHERE COMM BETWEEN 0.10 AND 0.12;

   e. **All customers excluding those with rating <= 100 unless they are located in Rome**
      - SELECT * FROM Customer WHERE RATING <= 100 AND CITY <> “Rome”;