***Module: 5(Database)***

**Q-1 What do you understand By Database?**

A database is an organized collection of structured information, or data, typically stored electronically in a computersystem. A database is usually controlled by a database management system (DBMS).

**Q-2 What is Normalization?**

**Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

**Q-3 What is Difference between DBMS and RDBMS?**

**ANS:** DBMS:

- DBMS does not support distributed database.

-DBMS stores data as file.

-Normalization is not present.

-It stores data in either a navigational or hierarchical form.

-No relationship between data.

-Examples: XML, Window Registry, etc.

RDBMS:

-RDBMS supports distributed database.

- RDBMS stores data in tabular form

- Normalization is present.

- It uses a tabular structure where the headers are the column names, and the rows contain corresponding values.

- Data is stored in the form of tables which are related to each other.

- Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

**Q-4 What is MF Cod Rule of RDBMS Systems?**

Rule 1**:** Information Rule: The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2**:** Guaranteed Access Rule: Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3**:** Systematic Treatment of NULL Values: The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog: The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule: A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6:View Updating Role: All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule: A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence: The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence: The logical data in a database must be independent of its user’s view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, thereshould be no impact or change on the user application.

Rule 10: Integrity Independence: A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Integrity Independence: A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 12: Non-Subversion Rule: If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

**Q-5 What do you understand By Data Redundancy?**

Data Redundancy means having multiple copies of same data in the database. This problem arises when a database is not normalized.

**Q-6 What is DDL Interpreter?**

It processes the DDL statements into a set of tables containing meta data (data about data).

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

It processes the DML statements into low level instruction (machine language), so that they can be executed.

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

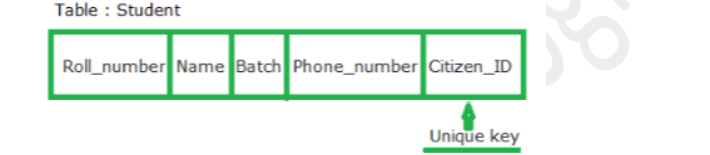
Unique Key: - Unique key constraints also identify an individual table uniquely in a relation or table.

- A table can have more than one unique key unlike primary key.

- Unique key constraints can accept only one NULL value for column.

- Unique constraints are also referenced by the foreign key of another table.

- It can be used when someone wants to enforce unique constraints on a column and a group of columns which is not a primary key.

****

Primary Key: - A primary key is a column of table which uniquely identifies each tuple (row) in that table.

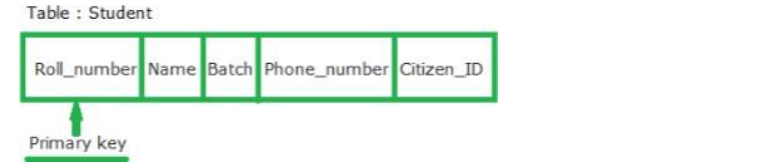
- Primary key enforces integrity constraints to the table.

- Only one primary key is allowed to use in a table.

- The primary key does not accept the any duplicate and NULL values.

- The primary key value in a table changes very rarely so it is chosen with care where the changes can occur in a seldom maner.

- A primary key of one table can be referenced by foreign key of another table.



Foreign Key: - When, "one" table's primary key field is added to a related "many" table in order to create the common field which relates the two tables, it is called a foreign key in the "many" table.

- In the example given below, salary of an employee is stored in salary table. Relation is established via is stored in "Employee" table. To identify the salary of "Jforeign key column ― Employee\_ID\_Ref‖ which refers ―Employee\_ID‖ field in Employee table of "Jhon" is stored in "Salary" table. But his employee info For example, salary hon", his "employee id" is stored with each salary record.



**Q-9 What is save Point? How to create a save Point write a Query?**

Savepoint: - 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.

- The syntax for a SAVEPOINT command is as shown below.

SAVEPOINT SAVEPOINT\_NAME;

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

Trigger: A trigger is a stored procedure in database which automatically invokeswhenever a special event in the database occurs.

- For example, a trigger can be invoked when a row is inserted into aspecified table.

- Syntax:

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

[trigger\_body]

***TASK***

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

create table student

(rollno int primary key,

name varchar(255),

branch varchar(255));

insert into student(rollno, name, branch)values

(1,"jay","computer science"),

(2,"suhani","electronic and comp"),

(3,"kriti","electronic and comp");

create table exam

(roll\_no int ,

foreign key(roll\_no) references student(rollno),

s\_code varchar(255),

marks int,

p\_code varchar(255));

insert into exam

(roll\_no,s\_code,marks,p\_code)values

(1,"cs11",50,"cs"),

(1,"cs12",60,"cs"),

(2,"ec101",66,"ec"),

(2,"cs102",70,"ec"),

(3,"ec101",45,"ec"),

(3,"ec102",50,"ec");

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

create table info

(First\_name varchar(255),

Last\_name varchar(255),

Address varchar(255),

City varchar(255),

age int);

insert into info

(First\_name,Last\_name,Address,City,age)values

("mickey", "mouse", "123 fantasy way","anaheim",73),

("bat","man","321 cavem ave","Gotham",54),

("wonder","woman","987Truth 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","itotiaw",28);

1. **Create table given below: Employee and Incentive**

Table name: Employee: -

create table Employee

(Employee\_id int primary key,

First\_name varchar(255),

Last\_name varchar(255),

salary bigint,

Joining\_date timestamp,

Department varchar(255));

insert into Employee

(Employee\_id,First\_name,Last\_name,salary,Joining\_date,Department)

values

(1 ,"john","abraham",1000000,"2013-01-01 12:00:00","Bankimg"),

(2 ,"Michael","Clarke",800000,"2013-01-01 12:00:00","Insurance"),

(3 ,"roy","thomas",700000,"2013-02-01 12:00:00","Bankimg"),

(4 ,"tom","jose",600000,"2013-02-01 12:00:00","insurance"),

(5 ,"jerry","pinto",650000,"2013-02-01 12:00:00","insurance"),

(6 ,"philip","mathew",750000,"2013-01-01 12:00:00","services"),

(7 ,"TestName1","123",650000,"2013-01-01 12:00:00","services"),

(8 ,"TestName2","Lname",600000,"2013-02-01 12:00:00","Insurance");

Table name: Incentive

create table incentive

(employee\_ref\_id int references employee(employee\_id),

incentive\_date date,

incentive\_amount int);

insert into incentive

(employee\_ref\_id,incentive\_date,incentive\_amount)values

(1,"2013-02-01",5000),

(2,"2013-02-01",3000),

(3,"2013-02-01",4000),

(1,"2013-01-01",4500),

(2,"2013-01-01",3500);

1. **Get First Name from employee table**

select First\_name from employee;

1. **Get FIRST\_NAME, Joining Date, and Salary from employee table.**

**ANS.** select First\_name,Joining\_date,salary from employee;

1. **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;

1. **Get employee details from employee table whose first name contains ‘J’.**

select \* from employee where First\_name like '%j%';

1. **Get department wise maximum salary from employee table order by salary ascending?**

select Department,max(salary) from employee group by Department order by salary asc;

1. **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 inner join incentive on employee.Employee\_id=incentive.incentive\_amount and incentive.incentive\_amount>3000;

1. **Create After Insert trigger on Employee table which insert records in view table**

create trigger tr

After insert

on employee

for each row

insert into view

(e\_id,f\_name,l\_name,salary,j\_date,department)values

(Employee\_id,First\_name,Last\_name,salary,Joining\_date,Department);

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

Table Name: Salesperson

create table salesperson

(Sno int primary key,

Sname varchar(255),

city varchar(255),

comm varchar(255));

insert into salesperson

(Sno,Sname,city,comm)values

(1001,"peel","London","0.12"),

(1002,"serres","san jose","0.13"),

(1004,"motika","london","0.11"),

(1007,"rafkin","barcelona","0.15"),

(1003,"Axelrod","new york","0.1");

Table Name: Customer

create table customer

(CNM int primary key,

CNAME varchar(255),

city varchar(255),

Rating int,

Sno int references salesperson(Sno));

insert into customer

(CNM,CNAME,city,Rating,Sno)values

(201, "Hoffman", "london", 100 , 1001),

(202, "Giovance", "roe", 200 , 1003),

(203, "liu", "san jose", 300 , 1002),

(204, "grass", "barcelona", 100 , 1002),

(206, "clemens", "london", 300 , 1007),

(207, "perreia", "roe", 100 , 1004);

1. **All orders for more than sno. 1003.**

Select \* from customer where sno>1003

1. **Names and cities of all salespeople in London with commission above 0.12**

select Sname,city from salesperson where city="london" and comm > 0.11;

1. **All salespeople either in Barcelona or in London**

select \* from salesperson where city=”london” or city=”barcelona”

1. **All salespeople with commission between 0.10 and 0.12. (Boundary values should be excluded).**

select \* from salesperson where comm > 0.1 and comm < 0.12;

1. **All customers excluding those with rating <= 100 unless they are located in Rome**

select \* from customer where rating<=100 and city=”Roe”