Assignment - II

1] Explain the following with example.

[a] Relationship types.

There are 3 types of relationships that can exist b/w two entities.

* One to one relationship: Such a relationship exists when each record of one table is related to only one of the other table.
  e.g: If there are two entities 'Person' (Id, name, age, address) & 'passport' (passport-id, passport-no) So each person can have only one passport & each passport belongs to only one person

* One to many OR Many to one: Such a relationship exists when each record of one table can be related to one or more than one record of the other table.
  e.g: If there are two entity type 'customer' & 'account' then each 'customer' can have more than one 'account' but each 'account' is held by only one 'customer'.

* Many to many relationship: Such a relationship exists when each record of the first table can be related to one or more than one record of the second table & a single record of the second table can be related to one or more than one record of the first table.
  e.g: If there are two entity type 'customer' & 'Product' then each customer can buy more than one product & a product can be bought by many diff. customer.

[b] Role, names & Recursive Relationship:

Each entity type that participates in a relationship type plays a particular role in the relationship.

The role name signifies the role that a participating entity from the entity type plays in each relationship instance. It helps to explain what the relationship means e.g: in the WORKS-FOR relationship type, EMPLOYEE plays the role of employee or worker & DEPARTMENT plays the role of department or employer.

In some cases the same ~~ety~~ entity type participate more than once in a relationship type in diff. roles in such cases the role name becomes essential for distinguishing the meaning of each participation. Such relationship types are called recursive relationship.

[i] Structural constraint: The constraints are determined from the miniworld situation that the relationship represent.

e.g: Suppose each employee must work for only one department.

2] Explain the characteristics of relations along with relational model notations. Also briefly describe relational model constraints.

Characteristics of Relations:

[I] Ordering of the relationship in tuple.

A relation is set of tuples. There is no particular order for elements in a set, Hence tuples in a relationship do not have for any particular order.

☒ Ordering of values within a tuple:

Means ordering of values associated with constraints in the relation R. The order of attributes & their values is not that important as long as the correspondance b/w attribute & values maintain alternative b/w attributes & values maintain alternatives.

☒ Values & Nulls in the tuple:

Each value in the tuple is an automic value ie not divisible into components. Therefore composit & multiple value attributes not allowed because of this property model is sometimes called flat relational model.

Relational model ~~constraints~~ Notations:

☒ following are the notations used for relational model.

R- Relational schema R ($A_1, A_2, A_3 \ldots A_n$) of degree N.

☒ The upper case letters Q. R. S denotes relation name.

☒ The lower case letters q, r, s. represents relation states

☒ t, u, v represents ~~ty~~ tuples.

Relational model constraints:

Constraints can be divided into 3 main categories.

① Model based: Constraints that are inherint in the data model are called as model based constraints.

eg: No duplicate tuples constraints.

② Schema based: Constraints that can be directly expressed in the schema of data model. which are typically specified by using DDL.

e.g: SSN- Primary key.

③ Application based: Constraints that cannot be expressed & hence must be expressed & imposed through appl'n programs.
e.g. Salary: employee ≯ manager.

3] Explain nested & co-related nested queries with example.
In nested queries a query is written inside a query. The result of inner query is used in execution of outer query. These nested queries are complete blocks SELECT-FROM-WHERE blocks within & another SQL.
The outer part queries called outer query & inner part queries called inner query.
The nested query can appear along with WHERE class or FROM class or SELECT class. Normally in nested query are the comparison operator 'IN' or 'EQUAL TO'
e.g. To find the names of employee who are dept. ID 103.

```
SELECT  E.ename.
FROM    Employee E
WHERE   E.id IN (SELECT D.id FROM dept. D. where D.ID =103).
```

Co-related nested queries are nested queries which are independent or depend only on the same row of an outer query being an embedded query.
e.g: To find the names of employee who are dept. id 103

```
SELECT  E.name
FROM    Employee E
WHERE   EXISTS (SELECT * FROM dept D. where D. id = 103
              AND D.id = E.id)
```

4) Explain with example EXISTS & NOT EXISTS functions in SQL.

The EXISTS fn in SQL is used to check whether the result of a co-related nested query is empty or not.

The result of EXISTS is a Boolean value TRUE if the nested query result containts atleast one tuple or FALSE if the nested query result constraint no tuples.

e.g Retrieve the name of each employee who has a dependent with the same first name & is the same sex as the employee.

```
SELECT L.FName, L.LName
FROM   EMPLOYEE AS E
WHERE  EXISTS ( select *
                FROM DEPENDENT AS D
                WHERE E.SSN= D.ESSN AND E.SEX = D.SEX
                AND E.FName = D.DependentName);
```

EXISTS & NOTEXISTS are typically used in conjunction with a co-related nested query.

e.g. Retrieve the names of employee who have no dependents

```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   NOT EXISTS (SELECT *
                    FROM DEPENDENT
                    WHERE Ssn= ESSN);
```

5) What are assertions & triggers in SQL ?
Write SQL program to create an assertion to specify the constraint that the salary of employee must not be greater than the salary of the department manager. The employee for in the COMPANY database

When a constraints involves 2 (or) more tables the table constraint mechanism is sometimes hard & results may not come as expected. To cover such situation in SQL supports the creation of assertions which are constraints not associated with only one table. And an assertion statement should ensure a certain condition will always exist in the database. DBMS always checks the assertion whenever modifications are done in the corresponding table.

```
CREATE   ASSERTIONS.   SALARY CONSTRAINTS
CHECK ( NOT EXISTS  (select *
                     from employee E1, employee m, dept. D.
                     where E salary > m. salary
                     AND  E.DNO = D.Number
                     AND  MGR.SSN = E.SSN));
```

A trigger is a database object that is associated with the table, it will be activated with the table it will be activated when a defined action is executed for the table. The trigger can be executed when we run the following statements
INSERT, UPDATE, DELETE
& it can be invoked before or after the event.

6) Write trigger in SQL to call a stored procedure. INFORM-SUPERVISOR() whenever a new record is created or updated. check whether the employee's salary is greater than the salary of his or her direct supervisor in the company database.

```
CREATE TRIGGER Salary_violation
BEFORE INSERT OR UPDATE of salary, supervisor_SSN
ON Employee for each row
when (NEW.salary > (SELECT salary
                    FROM employee
                    WHERE SSN = NEW. supervisor_SSN))
INFORM_SUPERVISOR (NEW_supervisor_SSN, NEW_SSN));
```

7) What are stored procedure in database applin development Explain the following w.r.to stored procedure.
a) Creating a simple stored procedure
b) Calling a stored procedure.

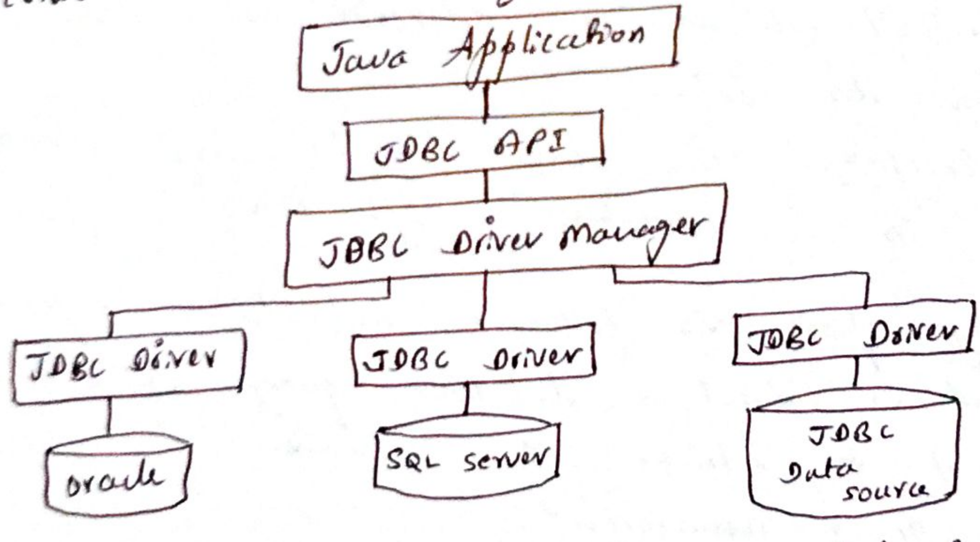
Stored procedures are created to perform one or more DML operations on database. It is nothing but the group of SQL statements that accepts some i/p in the form of parameters & performs some task & may or may not returns a value.

The stored procedure must have a name, this stored procedure has the name 'ShowNumberOfOrders' otherwise it just contains an SQL statement that is precompiled & stored at the server.

```
CREATE PROCEDURE ShowNumberOfOrders
SELECT C.cid , C.cname, COUNT (*)
    FROM   Customers C, orders O
    WHERE   C.cid = O.cid
    GROUP BY   C.cid , C.name.
```

Calling a stored procedure can be called in interactive SQL with the CALL statement.

8) Describe the architecture of JDBC.



<u>Drive manager</u>: This class manages a list of database drivers. Matches connection requests from the java applin with the proper database driver using communication sub protocol. The 1st driver that recognizes a certain subprotocol under JDBC will be used to establish a database connection.

<u>Driver</u>: This Interface handles the communication with the database server. Abstracts the details with working with driver objects.

Connection: This interface with all methods for contacting a database. The connection object represents communication context. ie all communication with database is through connection object only.

Statement: Some derived interface accept parameters in addition to executing stored procedures.

Resultset: These objects hold data retrieved from a database after ~~g~~ execute an SQL query using statement objects. It acts as an ~~intere~~ iterator to allow to move through its datas

SQL Exception: This class handles any errors that occur in a database application

9) Briefly explain the following individuals steps that we required to submit a database query to a data source & to retrieve the results.

JDBC Driver management:

- All drivers are managed by the Driver manager class.
- Loading a JDBC driver:

In the java code
class.forName ("Oracle / jdbc. driver. Oracle driver");
When starting the java appl'n
D-jdbc. drivers = oracle / jdbc. driver.

Connections:
Interacts with a data source through sessions. Each connection identifies a logical session.

JDBC URL:
jdbc;<subprotocol>: <other parameters>

Executing SQL statements:

Three diff. ways of executing SQL statements:

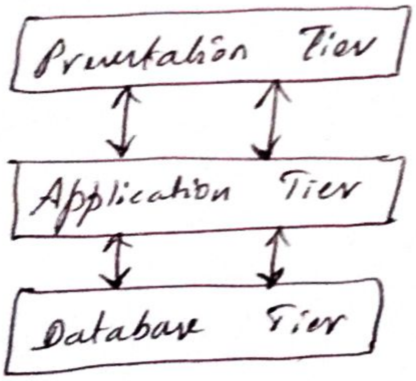
- Statement
- Prepared statement
- Callable Statement

Statements are both static & dynamic SQL statements.

Prepared statements are semi-static SQL statements.

Callable statements are stored procedures.

10) Describe Three-Tier architecture:



Database Tier: At this tier, the database resides along with its query processing languages. Also have the relations that define the data & their constraints at this level.

Application Tier: At this tier resides the appln server & the programs that access the database. For a user this appln tier represents an abstracted view of the database. End users are unaware of any existence of the database beyond the appln. At the other end the database tier is not aware of any other user beyond the appln tier. Hence the appln layer ~~sits~~ sits in the middle &

acts as a mediator b/w the end-user & the database.

User Tier: End users operate on this tier & they know nothing about any existence of the database beyond this layer. At this layer multiple views of the database can be provided by the application. All views are generated by applns that reside in the appln tier.