Laboratory 7

Title of the Laboratory Exercise: Nested queries and Join queries

1. **Introduction and Purpose of Experiment**

   Nesting of queries within another one is known as a nested queries. The query within another is known as a subquery. The statement containing a subquery is called a Parent Statement. The parent statement uses the rows returned by the subquery. SQL Join is used for combining column from two or more tables by using values common to both tables. Join Keyword is used in SQL queries. By doing this lab, students will be able to implement nested queries and join queries.

2. **Aim and Objectives**

   Aim

   - To design and implement nested queries and join queries using SQL commands

   Objectives

   At the end of this lab, the student will be able to

   - Design nested queries and join queries for the given problem statement
   - Execute the nested queries and join queries

3. **Experimental Procedure**

   - Analyse the problem statement
   - Create tables with appropriate attributes
   - Insert attribute values into the table
   - Design nested queries and join queries
   - Execute the SQL commands
   - Test the executed commands
   - Document the Results
   - Analyse and discuss the outcomes of your experiment

4. **Questions**

a. Create a tables for the given relational schema. Assume appropriate data type, and key constraints for each field.

Player (Name, Id, TeamNo, Score)

Team (TeamNo, TeamName)

b. Write the appropriate query for the following statements using SQL commands

i. Find the names of all the players who are in the same Team of 'Smith' (use nested query)

ii. Display the information about players who got Scores more than any player in TeamNo=1 (use nested query)

iii. Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (without join)

iv. Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (use join)

v. Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (use natural join)

vi. Display the players and their team names, in which the *TeamNo* is same in both the players and *Team* (use left outer join)

vii. Display the team names and the players involved, in which the *TeamNo* is same in both the players and *Team* (use right outer join)

c. Create suitable front end for querying and displaying the results

5. Presentation of Results

```
-- Create the respective tables
DROP TABLE IF EXISTS `PLAYER`;
DROP TABLE IF EXISTS `TEAM`;

CREATE TABLE TEAM
(
    TeamNo INT PRIMARY KEY,
    TeamName CHAR(20) NOT NULL
);

CREATE TABLE PLAYER
(
    Id INT PRIMARY KEY,
    Name CHAR(20),
    TeamNo INT,
    Score INT NOT NULL DEFAULT 0
);

ALTER TABLE PLAYER ADD FOREIGN KEY(TeamNo) REFERENCES TEAM(TeamNo);

-- Add data

INSERT INTO TEAM
VALUES
    (1, 'MI'),
    (2, 'RCB');

INSERT INTO PLAYER
VALUES
    (1, 'SMITH', 1, 50),
    (2, 'Deepak R', 1, 40),
    (3, 'Warner', 2, 35),
    (4, 'Virat', 2, 100);

-- View data

SELECT *
FROM PLAYER;

SELECT *
FROM TEAM;

-- Name of players who are in the same team as SMITH

SELECT Name
FROM PLAYER
WHERE TeamNo=(SELECT TeamNo
FROM PLAYER
WHERE NAME='SMITH');

-- players who got Scores more than any player in TeamNo=1
SELECT *
FROM PLAYER
WHERE Score > (SELECT MAX(Score)
FROM PLAYER
WHERE TeamNo=1);

-- players and team details in which TeamNo is same in both PLAYERS and TEAM (without join)
SELECT *
FROM PLAYER, TEAM
WHERE PLAYER.TeamNo = TEAM.TeamNo;

-- players and team details in which TeamNo is same in both PLAYERS and TEAM (with join)
SELECT *
FROM PLAYER JOIN TEAM ON PLAYER.TeamNo = TEAM.TeamNo;

-- players and team details in which TeamNo is same in both PLAYERS and TEAM (with natural join)
SELECT *
FROM PLAYER NATURAL JOIN TEAM;

-- players and team names in which TeamNo is same in both PLAYERS and TEAM (with left outer join)
SELECT *
FROM PLAYER LEFT OUTER JOIN TEAM ON PLAYER.TeamNo = Team.TeamNo;

-- team names and players involved in which TeamNo is same in both the PLAYERS and TEAM (with right outer join)
SELECT *
FROM PLAYER RIGHT OUTER JOIN TEAM ON PLAYER.TeamNo = Team.TeamNo;

```

Figure 0-0 Mysql Commands for given Problem

SELECT *FROM PLAYER ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching Rows:

#	Id	Name	TeamNo	Score
1	1	SMITH	1	50
2	2	Deepak R	1	40
3	3	Warner	2	35
4	4	Virat	2	100

Figure 0-1 PLAYER Table

SELECT *FROM TEAM ×

Page Size: 20 Total Rows: 2 Page: 1 of 1

#	TeamNo	TeamName
1	1	MI
2	2	RCB

Figure 0-2 TEAM Table

SELECT NameFROM PLAYERW... ×

#	Name
1	SMITH
2	Deepak R

Figure 0-3 Name of players who are in the same team as SMITH

SELECT *FROM PLAYERWHER... ×

Page Size: 20 Total Rows: 1 Page: 1 of 1 Matching Rows:

#	Id	Name	TeamNo	Score
1	4	Virat	2	100

Figure 0-4 players who got Scores more than any player in TeamNo=1

SELECT *FROM PLAYER, TEA... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching Rows:

#	Id	Name	TeamNo	Score	TeamNo	TeamName
1	1	SMITH	1	50	1	MI
2	2	Deepak R	1	40	1	MI
3	3	Warner	2	35	2	RCB
4	4	Virat	2	100	2	RCB

Figure 0-5 players and team details in which TeamNo is same in both PLAYERS and TEAM (without join)

SELECT *FROM PLAYER JOIN... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching Rows:

#	Id	Name	TeamNo	Score	TeamNo	TeamName
1	1	SMITH	1	50	1	MI
2	2	Deepak R	1	40	1	MI
3	3	Warner	2	35	2	RCB
4	4	Virat	2	100	2	RCB

Figure 0-6 players and team details in which TeamNo is same in both PLAYERS and TEAM (with join)

SELECT *FROM PLAYER NATU... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching Rows:

#	TeamNo	Id	Name	Score	TeamName
1	1	1	SMITH	50	MI
2	1	2	Deepak R	40	MI
3	2	3	Warner	35	RCB
4	2	4	Virat	100	RCB

Figure 0-7 players and team details in which TeamNo is same in both PLAYERS and TEAM (with natural join)

SELECT *FROM PLAYER LEFT... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching R

#	Id	Name	TeamNo	Score	TeamNo	TeamName
1	1	SMITH	1	50	1	MI
2	2	Deepak R	1	40	1	MI
3	3	Warner	2	35	2	RCB
4	4	Virat	2	100	2	RCB

Figure 0-8 players and team names in which TeamNo is same in both PLAYERS and TEAM (with left outer join)

SELECT *FROM PLAYER RIGH... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching Rows:

#	Id	Name	TeamNo	Score	TeamNo	TeamName
1	1	SMITH	1	50	1	MI
2	2	Deepak R	1	40	1	MI
3	3	Warner	2	35	2	RCB
4	4	Virat	2	100	2	RCB

Figure 0-9 team names and players involved in which TeamNo is same in both the PLAYERS and TEAM (with right outer join)

```
private void Button1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      try{
         Class.forName("com.mysql.jdbc.Driver");
         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/playerteam", "root", "root");
         Statement st = con.createStatement();
         ResultSet rs = st.executeQuery("SELECT Name FROM PLAYERWHERE TeamNo=(SELECT TeamNo FROM PLAYER
WHERE NAME='SMITH')");
         System.out.println("Name");

         while(rs.next()) {
            String Name = rs.getString("Name");
            String table[] ={Name};
            System.out.println("Name");
            DefaultTableModel t=(DefaultTableMadel)z.getModel();
            t.addRow(table);
```

Figure 0-10 represents java program to display teammates of smith which is encoded in Display Button

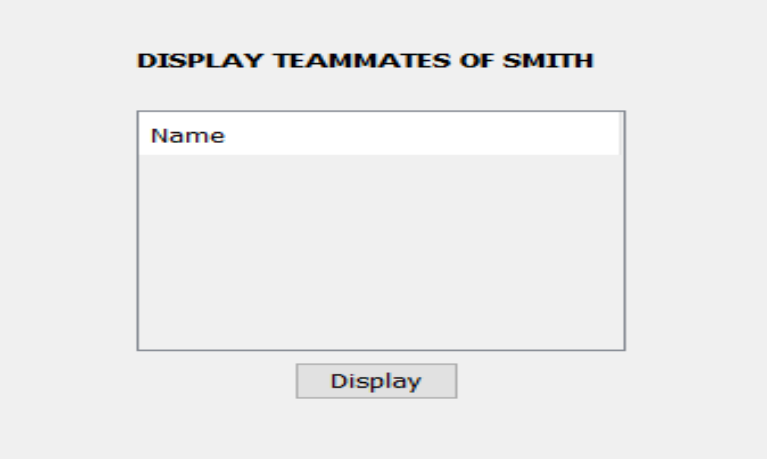

Figure0-11 Design of front end

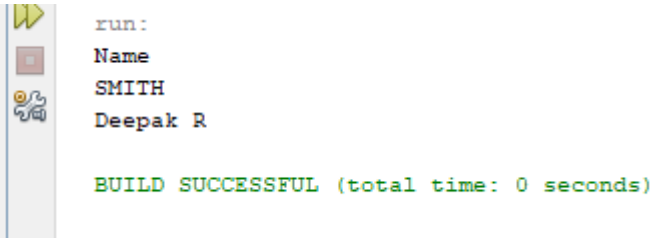

Figure 0-12 Output for the above java Program

6. Conclusions

A subquery can be nested inside other subqueries. SQL has an ability to nest queries within one another. A subquery is a SELECT statement that is nested within another SELECT statement and which return intermediate results. SQL executes innermost subquery first, then next level.

Important Rule:

- A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
- You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
- Subqueries are on the right side of the comparison operator.
- A subquery is enclosed in parentheses.
- In the Subquery, ORDER BY command cannot be used. But GROUP BY command can be used to perform the same function as ORDER BY command.

7. Comments

1. Limitations of Experiments

In general, you cannot modify a table and select from the same table in a subquery. For example, this limitation applies to statements of the following forms:

```
DELETE FROM t WHERE ... (SELECT ... FROM t ...);
UPDATE t ... WHERE col = (SELECT ... FROM t ...);
{INSERT|REPLACE} INTO t (SELECT ... FROM t ...);
```

2**. Limitations of Results**

- The subquery_select_list can consist of only one column name, except in the exists subquery, where an (*) is usually used in place of the single column name. You can use an asterisk (*) in a nested select statement that is not an exists subquery.
- You cannot use subqueries in an order by, group by, or compute by list.
- Subqueries cannot manipulate their results internally, that is, a subquery cannot include the order by clause, the compute clause, or the into keyword.
- Correlated (repeating) subqueries are not allowed in the select clause of an updatable cursor defined by declare cursor.
- You cannot include a union clause in a subquery unless it is part of a derived table expression within the subquery.

3**. Learning happened**

We learnt how to use nested queries and different types of joins in SQL