# 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.

1. 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

1. 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
2. Questions
   1. 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)

* 1. Write the appropriate query for the following statements using SQL commands
     1. Find the names of all the players who are in the same Team of ‘Smith’ (use nested query)
     2. Display the information about players who got Scores more than any player in TeamNo=1 (use nested query)
     3. Display the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (without join)
     4. Display the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use join)
     5. Display the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use natural join)
     6. Display the players and their team names, in which the *TeamNo* is same in both the playersand *Team* (use left outer join)
     7. Display the team names and the players involved, in which the *TeamNo* is same in both the playersand *Team* (use right outer join)
  2. Create suitable front end for querying and displaying the results

1. Presentation of Results

a)

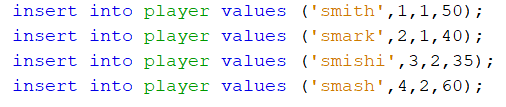


Fig: Inserting tuples into the player table

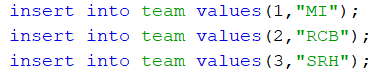


Fig: Inserting tuples into the team table

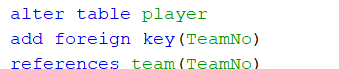


Fig: Declaring the foreign key

b)

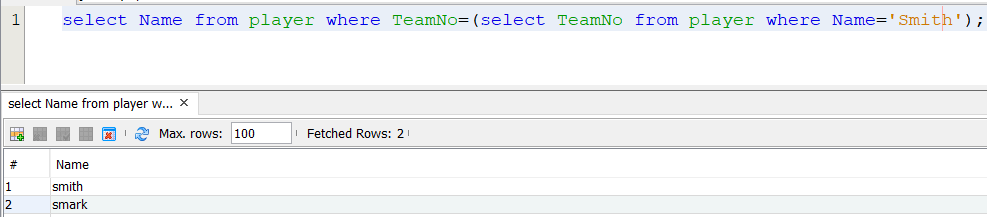


Fig: names of all the players who are in the same Team of ‘Smith’

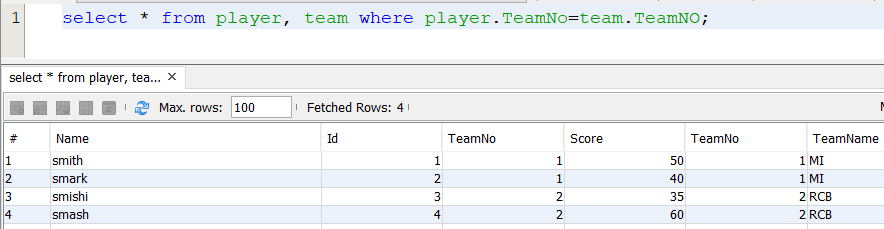


Fig: information about players who got Scores more than any player in TeamNo=1

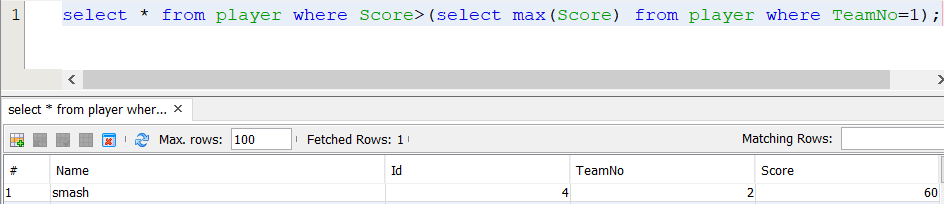


Fig: the players and Team details , in which the *TeamNo* is same in both the playersand *Team*

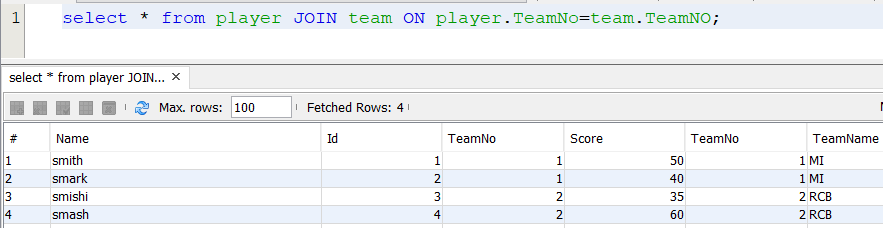


Fig: the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use join)

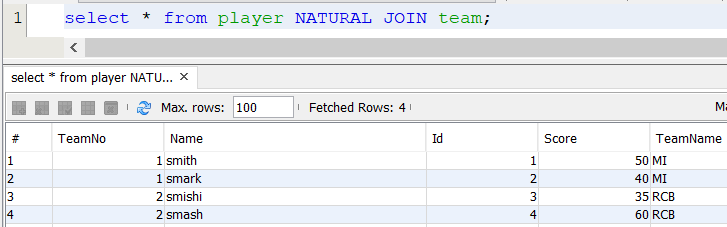


Fig: the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use natural join)

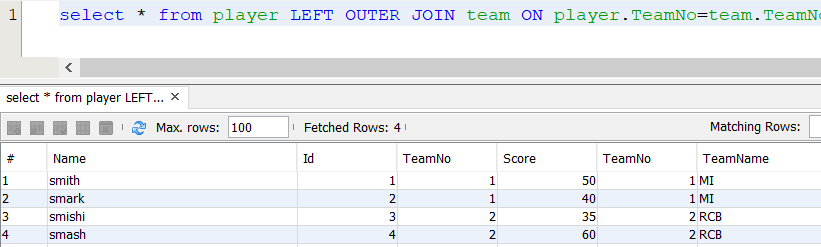


Fig: the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use left outerjoin)

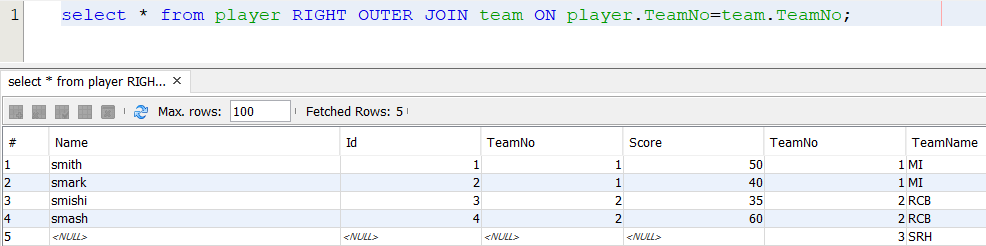


Fig: the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use right outer join)

1. Conclusions

1. Comments

1. Limitations of Experiments

2. Limitations of Results

3. Learning happened