Experiment No.6
Implement various join operations
Date of Performance:
Date of Submission:

Aim :- Write simple query to implement join operations(equi join, natural join, inner join, outer joins).

Objective :- To apply different types of join to retrieve queries from the database management system.

Theory:

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

SELECT

table1.column1,table1.column2,table2.

column1,.... FROM table1

RIGHT JOIN table2

ON table1.matching_column =

table2.matching_column; table1: First

table.

table2: Second table

matching_column: Column common to both the tables.

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

Syntax:

SELECT

table1.column1,table1.column2,table2.

column1,.... FROM table1

FULL JOIN table2

ON table1.matching_column =

table2.matching_column; table1:

First table.

table2: Second table

matching_column: Column common to both the tables.

Implementation:

```
SELECT Employee.EmployeeID, Employee.FirstName AS EmployeeFirstName,
Employee.LastName AS EmployeeLastName,

Manager.ManagerID, Manager.FirstName AS ManagerFirstName,
Manager.LastName AS ManagerLastName

FROM Employee

INNER JOIN Manager ON Employee.ManagerID = Manager.ManagerID;
```

```
SELECT Employee.EmployeeID, Employee.FirstName AS EmployeeFirstName,
Employee.LastName AS EmployeeLastName,

Manager.ManagerID, Manager.FirstName AS ManagerFirstName,
Manager.LastName AS ManagerLastName

FROM Employee

LEFT JOIN Manager ON Employee.ManagerID = Manager.ManagerID;
```

```
SELECT Employee.EmployeeID, Employee.FirstName AS EmployeeFirstName,
Employee.LastName AS EmployeeLastName,

Manager.ManagerID, Manager.FirstName AS ManagerFirstName,
Manager.LastName AS ManagerLastName

FROM Employee

RIGHT JOIN Manager ON Employee.ManagerID = Manager.ManagerID;
```

```
SELECT Employee.EmployeeID, Employee.FirstName AS EmployeeFirstName,
Employee.LastName AS EmployeeLastName,

Manager.ManagerID, Manager.FirstName AS ManagerFirstName,
Manager.LastName AS ManagerLastName

FROM Employee

FULL OUTER JOIN Manager ON Employee.ManagerID = Manager.ManagerID;
```

Conclusion:

1. Illustrate how to perform natural join for the joining attributes with different names with a suitable example.

Ans

In a natural join, tables are joined based on columns with the same name. If columns have different names, they can still be joined implicitly.

Example:

Two tables: "employees" and "departments".

"employees" has columns: emp_id, emp_name, dept_id.

"departments" has columns: dept_id, dept_name.

Performing a natural join:

```sql
SELECT *
FROM employees
NATURAL JOIN departments;
```

Result:

| emp_id | emp_name | dept_id | dept_name |

Even though the department ID columns have different names, the natural join automatically matches them based on their values.

2. Il lustrate significant differences between natural join equi join and inner join.

Ans

1. Natural Join:
   - Joins based on columns with the same name.
   - Automatically matches columns with identical names.
   - Can lead to unexpected results if column names are inconsistent.

2. Equi Join
   - Specific type of inner join using the equality operator (=).
   - Requires specifying columns to join on explicitly.
   - Returns rows where specified columns match.

3. Inner Join:
   - General join returning rows when there's a match based on specified conditions.
   - Can use any join condition, including equality or other logical conditions.
   - Requires explicitly specifying the join condition.

Key Differences:
- Natural join is based on column names, while equi join and inner join require explicit conditions.
- Equi join is a specific type of inner join using the equality operator.
- Inner join is a broader concept encompassing all joins where rows match based on specified conditions