Experiment No.6
Implement various joins and set operations
Name Of Student:-Bhagyashri Kaleni Sutar
Roll No:-75
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.

Conclusion:

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

Ans:-

A natural join typically relies on columns with the same name. However, if you have columns with *different* names but represent the same data, you need to use a variation of the natural join concept. Here's how you can achieve this with a suitable example, using standard SQL syntax (which may need slight adjustments depending on your specific database system):

Scenario:

Let's say you have two tables:

1. **`Employees`:**
    - `employee_id` (INT)
    - `employee_name` (VARCHAR)
    - `department_id` (INT)
2. **`Departments`:**
    - `dept_code` (INT)
    - `department_name` (VARCHAR)

You want to join these tables based on the department, but the columns are named differently (`department_id` in `Employees` and `dept_code` in `Departments`).

Solution (using `USING` or `ON` clause):

Since a true "natural join" won't work directly, you have two primary options:

1. **Using the `USING` clause (if your database supports it):**

    This clause is a shortcut for specifying the equality condition when the column names are different but represent the same value. However not all databases support the USING clause.

SELECT *FROM Employees

JOIN Departments USING (dept_code, department_id); -- This is wrong, and will not work as expected.

Important: The `USING` clause works correctly only when the column names are the same. Since this is not the case, the above example will not work.

Using the `ON` clause (most widely supported):

This is the most common and reliable method.

```
SELECT *
FROM Employees
JOIN Departments ON Employees.department_id = Departments.dept_code;
```

1. **Explanation:**
    - `JOIN Departments ON Employees.department_id = Departments.dept_code;`: This explicitly specifies the join condition, telling the database to match rows where the `department_id` from the `Employees` table equals the `dept_code` from the `Departments` table.
    - `SELECT *`: This retrieves all columns from both tables. You can customize this to select only the columns you need.

Example with Data:

```
-- Create Employees table
CREATE TABLE Employees (
    employee_id INT,
    employee_name VARCHAR(50),
    department_id INT
);

-- Create Departments table
CREATE TABLE Departments (
    dept_code INT,
    department_name VARCHAR(50)
);
```

```
-- Insert data into Employees
INSERT INTO Employees VALUES (1, 'Alice', 101);
INSERT INTO Employees VALUES (2, 'Bob', 102);
INSERT INTO Employees VALUES (3, 'Charlie', 101);
INSERT INTO Employees VALUES (4, 'David', 103);

-- Insert data into Departments
INSERT INTO Departments VALUES (101, 'Sales');
INSERT INTO Departments VALUES (102, 'Marketing');
INSERT INTO Departments VALUES (103, 'Engineering');

-- Perform the join
SELECT *
FROM Employees
JOIN Departments ON Employees.department_id = Departments.dept_code;
```

Result:

employee_id	employee_name	department_id	dept_code	department_name
1	Alice	101	101	Sales
2	Bob	102	102	Marketing
3	Charlie	101	101	Sales
4	David	103	103	Engineering

B) Illustrate significant differences between natural join equi join and inner join.

Ans:-

Feature	NATURAL JOIN	EQUI JOIN	INNER JOIN
Join Condition	Implicit (based on matching column names)	Explicit (equality operator `=` in `ON` clause)	Explicit (any condition in `ON` clause)
Syntax	`NATURAL JOIN`	`INNER JOIN ... ON table1.column = table2.column`	`INNER JOIN ... ON table1.column = table2.column` or other conditions
Flexibility	Limited (relies on matching column names)	Moderate (equality condition only)	High (any join condition)
Clarity	Potentially less clear (implicit)	Clear (explicit equality condition)	Clear (explicit condition)
Error Prone	More prone to errors (column name conflicts)	Less prone to errors	Less prone to errors
Database Support	Variable (not universally supported)	Universally supported (as part of `INNER JOIN`)	Universally supported
Example	`table1 NATURAL JOIN table2`	`table1 INNER JOIN table2 ON table1.id = table2.id`	`table1 INNER JOIN table2 ON table1.id = table2.id` or `table1 INNER JOIN table2 ON table1.id > table2.id`
Relationship	A type of Equi Join, which is a type of Inner Join	a type of Inner Join	The most general join, Equi and Natural are subsets