EXPERIMENT NO: 3

DATE: 03-Jun-2024

TITLE: To perform join operation between various tables.

OBJECTIVES:

- Fetch the data from more than one table on database.
- Know different types of join.

THEORY:

Join: A join is used when a SQL query requires data from more than one table on database.

There are two main types of join conditions: -

Equi-join

Non-equi join

Equi-join: The relationship between two tables is equi join when any one column corresponds to the same column in oyher table e.g. deptno in EMP table as well as in DEPT table. Here relationship is obtained using "="operator.

Non Equi-join: The relationship between two tables is non equi join when no column in one table corresponds directly to a column in other table. Here relationship is obtained other than "=" operator

Self Joins:

A self join is a join of a table to itself. This table appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition.

To perform a self join, Oracle combines and returns rows of the table that satisfy the join condition.

Inner Joins:

An inner join (sometimes called a "simple join") is a join of two or more tables that returns only those rows that satisfy the join condition.

Cross Joins:

If two tables in a join query have no join condition, Oracle returns their Cartesian product. Oracle combines each row of one table with each row of the other.

A Cartesian product always generates many rows and is rarely useful. For example, the Cartesian product of two tables, each with 100 rows, has 10,000 rows. Always include a join condition unless you specifically need a Cartesian product.

Outer Joins:

An outer join extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

- ➤ To write a query that performs an outer join of tables A and B and returns all rows from A (a left outer join), use the LEFT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of B in the join condition in the WHERE clause. For all rows in A that have no matching rows in B, Oracle returns null for any select list expressions containing columns of B.
 - > To write a query that performs an outer join of tables A and B and returns all

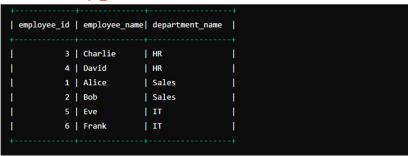
rows from B (a right outer join), use the RIGHT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of A in the join condition in the WHERE clause. For all rows in B that have no matching rows in A, Oracle returns null for any select list expressions containing columns of A.

> To write a query that performs an outer join and returns all rows from A and B, extended with nulls if they do not satisfy the join condition (a full outer join), use the FULL [OUTER] JOIN syntax in the FROM clause.

join), use the FULL [OUTER] JOIN syntax in the FROM clause. **EXCERCISE: Example Table:** Creating table and Inserting data. -- Creating the departments table **CREATE TABLE departments (** department_id INT PRIMARY KEY, department_name VARCHAR(50)); -- Creating the employees table **CREATE TABLE employees (** employee_id INT PRIMARY KEY, employee_name VARCHAR(50), department_id INT, salary DECIMAL(10, 2), job_title VARCHAR(50), manager_id INT, city VARCHAR(50), FOREIGN KEY (department_id) REFERENCES departments(department_id)); -- Inserting data into the departments table INSERT INTO departments (department_id, department_name) VALUES (1, 'Sales'), (2, 'HR'), (3, 'IT'); -- Inserting data into the employees table INSERT INTO employees (employee_id, employee_name, department_id, salary, job_title, manager_id, city) VALUES (1, 'Alice', 1, 12000, 'Manager', NULL, 'Baroda'), (2, 'Bob', 1, 9000, 'Salesman', 1, 'Ahmedabad'), (3, 'Charlie', 2, 11000, 'HR Executive', 4, 'Baroda'), (4, 'David', 2, 8000, 'Clerk', 5, 'Surat'), (5, 'Eve', 3, 13000, 'IT Manager', NULL, 'Baroda'), (6, 'Frank', 3, 9500, 'Developer', 5, 'Baroda');

- 1) Define: Join. Explain self join.
- 2) Retrieve employee number, employee name and their department name, in department name order.

SELECT e.emp_no, e.emp_name, d.dept_name FROM Employees e JOIN Departments d ON e.dept_no = d.dept_no ORDER BY d.dept_name;



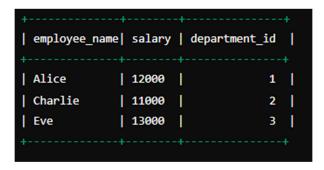
3) Show all employee details who lives in Baroda.

SELECT * FROM Employees WHERE city = 'Baroda';

```
employee_id | employee_name | department_id | salary | job_title
                                                                | manager_id | city
                                                                NULL
         1 | Alice
                                                  Manager
                                                                              Baroda
         3 | Charlie
                                                 | HR Executive | 4
                                                                            Baroda
                                                                            Baroda
         5 | Eve
                                       3 13000
                                                   IT Manager
                                                                NULL
         6 | Frank
                                                  Developer
                                                                            Baroda
```

4) Display the name, salary and department number of employees whose salary is more than 10000.

SELECT emp_name, salary, dept_no FROM Employees WHERE salary > 10000;



5) List the employee name, job, salary and department name for everyone in the company except clerks. Sort on salary displaying the highest salary first.

```
SELECT e.emp_name, e.job, e.salary, d.dept_name
FROM Employees e
JOIN Departments d ON e.dept_no = d.dept_no
WHERE e.job != 'Clerk'
ORDER BY e.salary DESC;
```



6) List all employees by name and number along with their manager's name and number.

SELECT e.emp_no, e.emp_name, m.emp_no AS mgr_no, m.emp_name AS mgr_name FROM Employees e

LEFT JOIN Employees m ON e.mgr_no = m.emp_no;

```
| employee_id | employee_name | manager_id | manager_name |
| 1 | Alice | NULL | NULL |
| 2 | Bob | 1 | Alice |
| 3 | Charlie | 4 | David |
| 4 | David | 5 | Eve |
| 5 | Eve | NULL | NULL |
| 6 | Frank | 5 | Eve |
```

7) Display all the employees who earn less than their managers.

```
SELECT e.emp_no, e.emp_name, e.salary, m.emp_no AS mgr_no, m.salary AS mgr_salary
FROM Employees e
JOIN Employees m ON e.mgr_no = m.emp_no
WHERE e.salary < m.salary;
```