Lab 04: Data retrieval operations in SQL using join operations.

**Retrieving Data from Multiple Tables**

In our previous lab session, we learned how to get data from a single table. But often, we need to get data from more than one table at the same time. For example, if we want a report that shows:

* Employee ID
* Employee Name
* Job Title
* Department Name

The first three pieces of information (Employee ID, Name, and Job) are found in the **EMP** table, while the Department Name is found in the **DEPT** table. To create this report, we need to connect (or link) the **EMP** and **DEPT** tables together. This linking process is called a **join operation**.

**What is a Join Operation?**

A **join operation** allows us to combine data from multiple tables into one result set based on a related column between the tables.

**Cartesian Product**

To understand joins better, we first need to know about the **Cartesian Product**. Here’s what it means:

* A **Cartesian Product** happens when every row from the first table is combined with every row from the second table.
* This can occur in two situations:
  1. When we don’t include any conditions for joining the tables.
  2. When the conditions we use for joining the tables are incorrect.

**Example of Cartesian Product**

Imagine we have two tables:

* **EMP** table has 14 rows (employees).
* **DEPT** table has 4 rows (departments).

If we run this query:

sql

SELECT \*

FROM EMP, DEPT;

The result will be a combination of every employee with every department, which means:

* 14 employees x 4 departments = **56 rows** in total.

This is an example of a Cartesian Product because we didn’t specify any conditions to limit the results.

**Standard Format for Cartesian Product**

There’s a special way to write a query for a Cartesian Product using **CROSS JOIN**:

sql

SELECT \*

FROM EMP CROSS JOIN DEPT;

**Why is a Cartesian Product Rarely Useful?**

A Cartesian Product usually results in a very large number of rows, and the information it provides is often not helpful. That’s why it’s important to include valid conditions when we join tables.

* A **join** is basically a smaller, more useful result set that comes from a Cartesian Product but with specific conditions applied.

**Types of Joins**

When we combine data from different tables, we use different types of joins. Each type of join works a bit differently and is useful in different situations. Oracle 9i introduced a new way to write joins that follows the SQL standard, but it doesn't make queries faster than the old way.

**1. Inner Join / Equi-Join**

* **Inner Join** (also called **Equi-Join**) combines rows from two tables where there is a match based on a common column.

**Example:** To get the employee name, their job, and the department name, we need to pull information from both the **EMP** table and the **DEPT** table. We do this by making sure the **DEPTNO** (Department Number) in both tables is equal.

Here’s how you can write that query:

sql

SELECT E.ENAME, E.JOB, D.DNAME

FROM EMP E, DEPT D

WHERE E.DEPTNO = D.DEPTNO;

* The SQL standard also allows you to use a simpler way to write this:

sql

SELECT ENAME, JOB, DNAME

FROM EMP NATURAL JOIN DEPT;

**2. Outer Join**

An **Outer Join** includes all the results from the inner join and also shows unmatched rows from one or both tables. There are three types of outer joins:

* **Left Outer Join**: Includes all records from the left table and matched records from the right table.

**Example:**

sql

SELECT E.ENAME, D.DEPTNO, D.DNAME

FROM EMP E, DEPT D

WHERE E.DEPTNO = D.DEPTNO(+);

* Using the SQL standard, you can write it this way:

sql

SELECT E.ENAME, D.DEPTNO, D.DNAME

FROM EMP E LEFT OUTER JOIN DEPT D

ON (E.DEPTNO = D.DEPTNO);

* **Right Outer Join**: Includes all records from the right table and matched records from the left table.

**Example:**

sql

SELECT E.ENAME, D.DEPTNO, D.DNAME

FROM EMP E, DEPT D

WHERE E.DEPTNO(+) = D.DEPTNO;

* Using the SQL standard:

sql

SELECT E.ENAME, D.DEPTNO, D.DNAME

FROM EMP E RIGHT OUTER JOIN DEPT D

ON (E.DEPTNO = D.DEPTNO);

* **Full Outer Join**: Combines the results of both left and right outer joins. It shows all records from both tables.

**Example:**

sql

SELECT E.ENAME, D.DEPTNO, D.DNAME

FROM EMP E FULL OUTER JOIN DEPT D

ON (E.DEPTNO = D.DEPTNO);

**3. Non-Equi Join**

A **Non-Equi Join** uses an inequality instead of an equality to join tables. This type of join is useful when you want to retrieve data based on ranges.

**Example:** To get the employee name, their salary, and their grade from two tables (EMP and SALGRADE):

sql

SELECT E.ENAME, E.SAL, S.GRADE

FROM EMP E, SALGRADE S

WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL;

**4. Self Join**

A **Self Join** is when you join a table to itself. This is helpful when you need to compare rows within the same table, such as finding out who each employee’s manager is.

**Example:** To find out which employee works for which manager:

sql

SELECT WORKER.ENAME || ' works for ' || MANAGER.ENAME

FROM EMP WORKER, EMP MANAGER

WHERE WORKER.MGR = MANAGER.EMPNO;

**Summary**

* **Inner Join/Equi-Join**: Combines rows with matching values in both tables.
* **Outer Join**: Includes unmatched rows from one or both tables.
  + Left Outer Join: All from the left, matched from the right.
  + Right Outer Join: All from the right, matched from the left.
  + Full Outer Join: All from both tables.
* **Non-Equi Join**: Uses inequalities to join tables.
* **Self Join**: Joins a table to itself to compare data within the same table.

These joins help us get the information we need from multiple tables effectively! If you have any questions or need more details, just let me know