

**USMAN INSTITUTE OF TECHNOLOGY**

**Department of Computer Science**  
**CS311 Introduction to Database Systems**

**Lab#4**

**Objective:**

Data retrieval operations in SQL using join operations.

Name of Student: Muhammad Waleed

Roll No: 20B-115-SE      Sec. B

Date of Experiment: \_\_\_\_\_

.....

Marks Obtained/Remarks: \_\_\_\_\_

Signature: \_\_\_\_\_

## THEORY

In previous lab sessions, we learned different ways to retrieve data from a single table. However, we frequently need data from more than one table. For example, suppose we need a report that displays employee id, name, job and department name. The first three attributes are present in EMP table where as the last one is in DEPT table (see lab session 1). To produce the report, we need to link the EMP and DEPT tables and access data from both of them. This is called join operation. To gain better understanding of join, it would be helpful to first clarify the concept of Cartesian Product.



Figure 4.1: The join operation collects data from multiple sources

## Cartesian Product

A Cartesian Product results when all rows in the first table are joined to all rows in the second table. A Cartesian product is formed under following conditions: -

- i. When a join condition is omitted
- ii. When a join condition is invalid

Consider the following example: -

```
SELECT *  
FROM EMP, DEPT;
```

In the above example, if EMP table has 14 rows and DEPT table has 4 rows, then their Cartesian product would generate  $14 \times 4 = 56$  rows. In fact, the ISO standard provides a special format of the SELECT statement for the Cartesian product: -

```
SELECT *  
FROM EMP CROSS JOIN DEPT;
```

A Cartesian product tends to generate a large number of rows and its result is rarely useful. It is always necessary to include a valid join condition in a WHERE clause. Hence a join is always a subset of a Cartesian product.

## Types of Joins

There are various forms of join operation, each with subtle differences, some more useful than others. The Oracle 9i database offers join syntax that is SQL 1999 compliant. Prior to release 9i, the join syntax was different from the ANSI standard. However, the new syntax does not offer any performance benefits over the Oracle proprietary join syntax that existed in prior releases.

**i. Inner-Join/Equi-Join**

If the join contains an equality condition, it is called equi-join.

**Example:**

To retrieve the employee name, their job and department name, we need to extract data from two tables, EMP and DEPT. This type of join is called *equijoin*-that is, values in the DEPTNO column on both tables must be equal. Equijoin is also called *simple join* or *inner join*.

```
SELECT E.ENAME, E.JOB, D.DNAME
FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO;
```

The SQL-1999 standard provides the following alternative ways to specify this join:-

```
SELECT ENAME, JOB, DNAME
FROM EMP NATURAL JOIN DEPT;
```

**ii. Outer-Join**

A join between two tables that returns the results of the inner join as well as unmatched rows in the left or right tables is a left or right outer join respectively. A full outer join is a join between two tables that returns the results of a left and right join.

**Left Outer Join****Example:**

```
SELECT E.ENAME, D.DEPTNO, D.DNAME
FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO(+);
```

**NOTE:** The outer join operator appears on only that side that has information missing.

The SQL-1999 standard provides the following alternative way to specify this join: -

```
SELECT E.ENAME, D.DEPTNO, D.DNAME
FROM EMP E LEFT OUTER JOIN DEPT D
ON (E.DEPTNO = D.DEPTNO);
```

**Right Outer Join****Example:**

```
SELECT E.ENAME, D.DEPTNO, D.DNAME
FROM EMP E, DEPT D
WHERE E.DEPTNO(+) = D.DEPTNO;
```

The SQL-1999 standard provides the following alternative way to specify this join: -

```
SELECT E.ENAME, D.DEPTNO, D.DNAME
FROM EMP E RIGHT OUTER JOIN DEPT D
ON (E.DEPTNO = D.DEPTNO);
```

**NOTE:** In the equi-join condition of EMP and DEPT tables, department OPERATIONS does not appear because no one works in that department. In the outer join condition, the OPERATIONS department also appears. The output is shown in Figure 4.2.

ENAME	DEPTNO	DNAME
CLARK	10	ACCOUNTING
KING	10	ACCOUNTING
ARMANO	10	ACCOUNTING
MILLER	10	ACCOUNTING
SMITH	20	RESEARCH
ADAMS	20	RESEARCH
FORD	20	RESEARCH
SCOTT	20	RESEARCH
JONES	20	RESEARCH
ALLEN	30	SALES
BLAKE	30	SALES
MARTIN	30	SALES
JAMES	30	SALES
TURNER	30	SALES
WARD	30	SALES
	40	OPERATIONS

16 rows selected.

Figure 4.2: Joining tables using right outer-join

### Full Outer Join

The SQL-1999 standard provides the following way to specify this join:-

```
SELECT E.ENAME, D.DEPTNO, D.DNAME
FROM EMP E FULL OUTER JOIN DEPT D
ON (E.DEPTNO = D.DEPTNO);
```

### iii. Non-Equijoin

If the join contains inequality condition, it is called non-equijoin. E.g. to retrieve employee name, salary and their grades using *non-equijoins*, we need to extract data from two tables,

```
EMP and SALGRADE.
SELECT E.ENAME, E.SAL, S.GRADE
FROM EMP E, SALGRADE S
WHERE E.SAL
BETWEEN S.LOSAL AND S.HISAL;
```

### iv. Self Join

To find the name of each employee's manager, we need to join the EMP table to itself, or perform a *self join*.

```
SELECT WORKER.ENAME || ' works for ' || MANAGER.ENAME
FROM EMP WORKER, EMP MANAGER
WHERE WORKER.MGR = MANAGER.EMPNO;
```

## EXERCISES

- i. To display the employee name, department name, and location of all employees who earn a commission.

```
Select Ename, Dname, Loc
From Emp Natural Join Dept
Where Comm Is Not Null;
```

	ENAME	DNAME	LOC
1	ALLEN	SALES	CHICAGO
2	TURNER	SALES	CHICAGO
3	MARTIN	SALES	CHICAGO
4	WARD	SALES	CHICAGO

- ii. To display all the employee's name (including KING who has no manager) and their manager name.

```
Select Worker.Ename As "Employee",
Manager.Ename As "Manger"
From Emp Worker, Emp Manager
Where Worker.Mgr = Manager.Empno(+);
/*Question 3*/
```

	Employee	Manger
1	FORD	JONES
2	SCOTT	JONES

- iii. To display the name of all employees whose manager is **KING**.

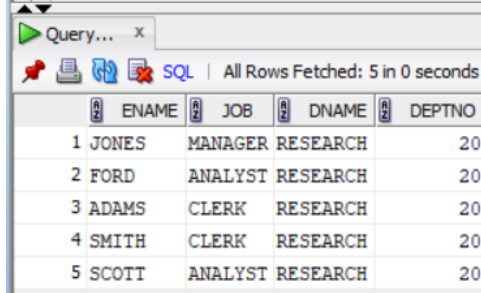
```
Select Worker.Ename
From Emp Worker , Emp Manager
Where Worker.Mgr = Manager.Empno
And Manager.Ename = 'KING';
```

	ENAME
1	BLAKE
2	JONES
3	CLARK

## Lab No. 4

- iv. Write a query to display the name, job, department number and department name for all employees who work in DALLAS.

```
Select Ename, Job, Dname, Deptno
From Emp Natural Join Dept
Where Loc='DALLAS';
```

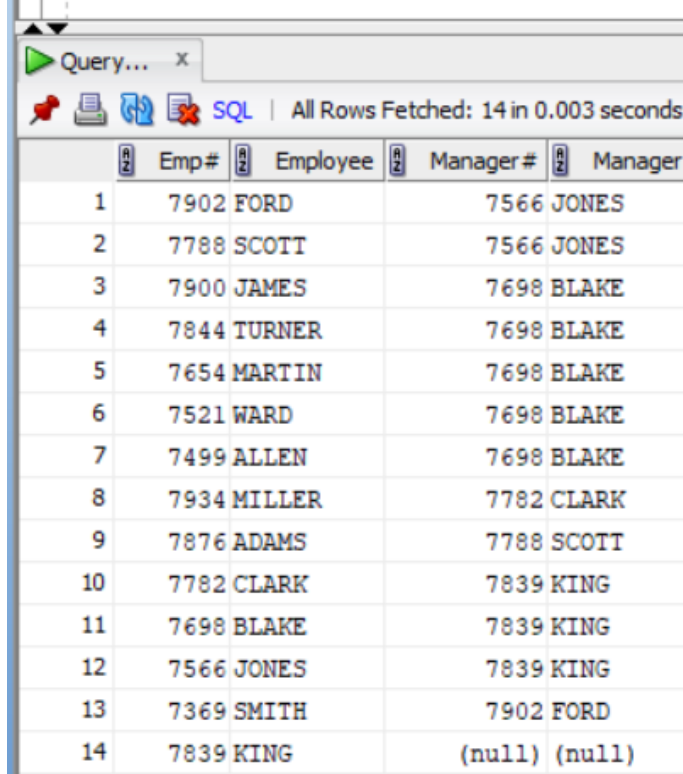


The screenshot shows a SQL query window with the query: `Select Ename, Job, Dname, Deptno From Emp Natural Join Dept Where Loc='DALLAS';`. The results are displayed in a table with 5 rows and 4 columns: ENAME, JOB, DNAME, and DEPTNO. The data is as follows:

	ENAME	JOB	DNAME	DEPTNO
1	JONES	MANAGER	RESEARCH	20
2	FORD	ANALYST	RESEARCH	20
3	ADAMS	CLERK	RESEARCH	20
4	SMITH	CLERK	RESEARCH	20
5	SCOTT	ANALYST	RESEARCH	20

- v. Display the employee name and employee number along with their manager's name Manager Number. Label the columns Employee, Emp#, Manager, and Manager#, respectively.

```
Select Worker.Empno As "Emp#",
Worker.Ename As "Employee",
Manager.Empno As "Manager#" ,
Manager.Ename As "Manager"
From Emp Worker, Emp Manager
Where Worker.Mgr = Manager.Empno(+);
```



The screenshot shows a SQL query window with the query: `Select Worker.Empno As "Emp#", Worker.Ename As "Employee", Manager.Empno As "Manager#" , Manager.Ename As "Manager" From Emp Worker, Emp Manager Where Worker.Mgr = Manager.Empno(+);`. The results are displayed in a table with 14 rows and 4 columns: Emp#, Employee, Manager#, and Manager. The data is as follows:

	Emp#	Employee	Manager#	Manager
1	7902	FORD	7566	JONES
2	7788	SCOTT	7566	JONES
3	7900	JAMES	7698	BLAKE
4	7844	TURNER	7698	BLAKE
5	7654	MARTIN	7698	BLAKE
6	7521	WARD	7698	BLAKE
7	7499	ALLEN	7698	BLAKE
8	7934	MILLER	7782	CLARK
9	7876	ADAMS	7788	SCOTT
10	7782	CLARK	7839	KING
11	7698	BLAKE	7839	KING
12	7566	JONES	7839	KING
13	7369	SMITH	7902	FORD
14	7839	KING	(null)	(null)

