Database Systems LAB # 12

Relating Data through Join Concept in Oracle RDBMS

Joins:

The purpose of a join concept is to combine data spread across tables. A join is actually performed by the 'where' clause which combines specified rows of tables.

Types of Joins

1. Cross Join
2. Inner Joins
3. Outer Joins

1. Cross Join

A cross join performs a relational product (also known as the Cartesian product) of two tables. The cross join syntax is:

SELECT *column-list* FROM *table1* CROSS JOIN *table2*

For Example:

*select * from test3 cross join emp;*

2. Inner Joins

a. Natural Join

Natural join returns all rows with matching values in the matching columns and eliminates duplicate columns. This style of query is used when the tables share one or more common attributes with common names.

The natural join syntax is:

SELECT *column-list* FROM *table1* NATURAL JOIN *table2*

Example:
*select * from emp natural join test3;*

The natural join will perform the following tasks:

- Determine the common attribute(s) by looking for attributes with identical names and compatible data types.
- Select only the rows with common values in the common attribute.
- If there are no common attributes, return the relational product of the two tables.

b. Join with Using Clause

A second way to express a join is through the USING keyword. The query returns only the rows with matching values in the column indicated in the USING clause—and that column must exist in both tables.

The syntax is:

SELECT *column-list* FROM *table1* JOIN *table2* USING (*common-column*)

Example :

*select * from emp join test3 using (job);*

c. Join with ON Clause

The previous two join styles use common attribute names in the joining tables. Another way to express a join when the tables have no common attribute names is to use the JOIN ON operand. The query will return only the rows that meet the indicated join condition. The join condition will typically include an equality comparison expression of two columns. (The columns may or may not share the same name, but obviously they must have comparable data types.)

The syntax is:
SELECT *column-list* FROM *table1* JOIN *table2* ON *join-condition*

Example:

*select * from emp x join test3 y on (x.Ename = y.Name);*

3. Outer Joins

An outer join returns not only the rows matching the join condition (that is, rows with matching values in the common columns), it returns the rows with unmatched values.

a. **Left Outer Join**

The left outer join returns not only the rows matching the join condition, it returns the rows in the left table with unmatched values in the right table.

The syntax is:
SELECT *column-list* FROM *table1* LEFT JOIN *table2* ON *join-condition*

Example:

*select * from emp x left join test3 y using (Job);*

b. **Right Outer Join**

The right outer join returns not only the rows matching the join condition, it returns the rows in the right table with unmatched values in the left table.

The syntax is:

SELECT *column-list* FROM *table1* RIGHT JOIN *table2* ON *join-condition*

*select * from emp x right join test3 y using (Job);*

c. **Full Outer Join**

The full outer join returns not only the rows matching the join condition (that is, rows with matching values in the common column), it returns all of the rows with unmatched values in the table on either side.

The syntax is:
SELECT *column-list* FROM *table1* FULL JOIN *table2* ON *join-condition*

Example:

*select * from emp x full join test3 y using (Job);*