GET DATA FROM MULTIPLE TABLES

Displaying Data from Multiple Tables

The related tables of a large database are linked through the use of foreign and primary keys or what are often referred to as common columns. The ability to join tables will enable you to add more meaning to the result table that is produced. For 'n' number tables to be joined in a query, minimum $n-1$ join conditions are necessary. Based on the join conditions, Oracle combines the matching pair of rows and displays the one which satisfies the join condition.

Joins are classified as below

- Natural join *alsoknownasanequijoinorasimplejoin* - Creates a join by using a commonly named and defined column.
- Non-equality join - Joins tables when there are no equivalent rows in the tables to be joined-for example, to match values in one column of a table with a range of values in another table.
- Self-join - Joins a table to itself.
- Outer join - Includes records of a table in output when there's no matching record in the other table.
- Cartesian join *alsoknownasaCartesianproductorcrossjoin* - Replicates each row from the first table with every row from the second table.Creates a join between tables by displaying every possible record combination.

Natural Join

The NATURAL keyword can simplify the syntax of an equijoin.A NATURAL JOIN is possible whenever two *ormore* tables have columns with the same name,and the columns are join compatible, i.e., the columns have a shared domain of values.The join operation joins rows from the tables that have equal column values for the same named columns.

Consider the one-to-many relationship between the DEPARTMENTS and EMPLOYEES tables.Each table has a column named DEPARTMENT_ID.This column is the primary key of the DEPARTMENTS table and a foreign key of the EMPLOYEES table.

```
SELECT E.first_name NAME,D.department_name DNAME
FROM employees E NATURAL JOIN departments D;

FIRST_NAME DNAME
---------- ------
MILLER     DEPT 1
JOHN       DEPT 1
MARTIN     DEPT 2
EDWIN      DEPT 2
```

The below SELECT query joins the two tables by explicitly specifying the join condition with the ON keyword.

```
SELECT E.first_name NAME,D.department_name DNAME
FROM employees E JOIN departments D
ON (E.department_id = D.department_id);
```

There are some limitations regarding the NATURAL JOIN.You cannot specify a LOB column with a NATURAL JOIN.Also, columns involved in the join cannot be qualified by a table name or alias.

USING Clause

Using Natural joins, Oracle implicitly identify columns to form the basis of join. Many situations

require explicit declaration of join conditions. In such cases, we use USING clause to specify the joining criteria. Since, USING clause joins the tables based on equality of columns, it is also known as Equijoin. They are also known as Inner joins or simple joins.

Syntax:

```
SELECT <column list>
FROM   TABLE1   JOIN   TABLE2
USING (column name)
```

Consider the below SELECT query, EMPLOYEES table and DEPARTMENTS table are joined using the common column DEPARTMENT_ID.

```
SELECT E.first_name NAME,D.department_name DNAME
FROM employees E JOIN departments D
USING (department_id);
```

Self Join

A SELF-JOIN operation produces a result table when the relationship of interest exists among rows that are stored within a single table. In other words, when a table is joined to itself, the join is known as Self Join.

Consider EMPLOYEES table,which contains employee and their reporting managers.To find manager's name for an employee would require a join on the EMP table itself. This is a typical candidate for Self Join.

```
SELECT e1.FirstName Manager,e2.FirstName Employee
FROM employees e1 JOIN employees e2
ON (e1.employee_id = e2.manager_id)
ORDER BY e2.manager_id DESC;
```

Non Equijoins

A non-equality join is used when the related columns can't be joined with an equal sign-meaning there are no equivalent rows in the tables to be joined.A non-equality join enables you to store a range's minimum value in one column of a record and the maximum value in another column. So instead of finding a column-tocolumn match, you can use a non-equality join to determine whether the item being shipped falls between minimum and maximum ranges in the columns.If the join does find a matching range for the item, the corresponding shipping fee can be returned in the results. As with the traditional method of equality joins, a non-equality join can be performed in a WHERE clause. In addition, the JOIN keyword can be used with the ON clause to specify relevant columns for the join.

```
SELECT E.first_name,
            J.job_hisal,
            J.job_losal,
            E.salary
     FROM employees E JOIN job_sal J
     ON (E.salary BETWEEN J.job_losal AND J.job_losal);
```

We can make use all comparison parameter discussed earlier like equality and inequality operators, BETWEEN, IS NULL, IS NOT NULL, and RELATIONAL.

Outer Joins

An Outer Join is used to identify situations where rows in one table do not match rows in a second table, even though the two tables are related.

There are three types of outer joins: the LEFT, RIGHT, and FULL OUTER JOIN. They all begin with an INNER JOIN, and then they add back some of the rows that have been dropped. A LEFT OUTER JOIN adds back all the rows that are dropped from the first *left* table in the join condition, and output columns from the second *right* table are set to NULL. A RIGHT OUTER JOIN adds back all the rows that are dropped from the second *right* table in the join condition, and output columns from the first

left table are set to NULL. The FULL OUTER JOIN adds back all the rows that are dropped from both the tables.

Right Outer Join

A RIGHT OUTER JOIN adds back all the rows that are dropped from the second *right* table in the join condition, and output columns from the first *left* table are set to NULL. Note the below query lists the employees and their corresponding departments. Also no employee has been assigned to department 30.

```
SELECT E.first_name, E.salary, D.department_id
FROM employees E, departments D
WHERE E.DEPARTMENT_ID (+) = D.DEPARTMENT_ID;

FIRST_NAME SALARY     DEPARTMENT_ID
---------- ---------- ----------
JOHN       6000       10
EDWIN      2000       20
MILLER     2500       10
MARTIN     4000       20
                      30
```

Left Outer Join

A LEFT OUTER JOIN adds back all the rows that are dropped from the first *left* table in the join condition, and output columns from the second *right* table are set to NULL. The query demonstrated above can be used to demonstrate left outer join, by exchanging the position of `+` sign.

```
SELECT E.first_name, E.salary, D.department_id
FROM employees E, departments D
WHERE   D.DEPARTMENT_ID = E.DEPARTMENT_ID (+);

FIRST_NAME SALARY     DEPARTMENT_ID
---------- ---------- ----------
JOHN       6000       10
EDWIN      2000       20
MILLER     2500       10
MARTIN     4000       20
                      30
```

Full Outer Join

The FULL OUTER JOIN adds back all the rows that are dropped from both the tables. Below query shows lists the employees and their departments. Note that employee 'MAN' has not been assigned any department till now *itsNULL* and department 30 is not assigned to any employee.

```
SELECT  nvl (e.first_name,'-') first_name, nvl (to_char (d.department_id),'-')
department_id
FROM employee e FULL OUTER JOIN department d
ON e. depARTMENT_ID = d. depARTMENT_ID;

FIRST_NAME DEPARTMENT_ID
---------- --------------------
MAN        -
JOHN       10
EDWIN      20
MILLER     10
MARTIN     20
-          30

6 rows selected.
```

Cartesian product or Cross join

For two entities A and B, A * B is known as Cartesian product. A Cartesian product consists of all possible combinations of the rows from each of the tables. Therefore, when a table with 10 rows is

joined with a table with 20 rows, the Cartesian product is 200 rows $10 * 20 = 200$.For example, joining the employee table with eight rows and the department table with three rows will produce a Cartesian product table of 24 rows $8 * 3 = 24$.

Cross join refers to the Cartesian product of two tables. It produces cross product of two tables. The above query can be written using CROSS JOIN clause.

A Cartesian product result table is normally not very useful. In fact, such a result table can be terribly misleading. If you execute the below query for the EMPLOYEES and DEPARTMENTS tables, the result table implies that every employee has a relationship with every department, and we know that this is simply not the case!

```
SELECT E.first_name, D.DNAME
FROM employees E,departments D;
```

Cross join can be written as,

```
SELECT E.first_name, D.DNAME
FROM employees E CROSS JOIN departments D;
```