# JOINS

## Let Us Understand JOINS in Oracle

### Retrieving Data From More Than One Table's

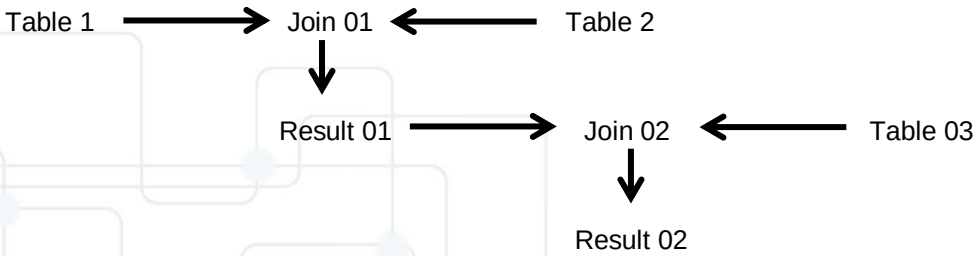
### JOINS:

- A Join is a Query That Combines Rows From Two or More Tables, Views, or Materialized Views.
- A Join is Performed Whenever Multiple Tables Appear in The Queries FROM Clause.
- The Queries SELECT List Can Select Any Columns From Any of These Tables.
- The Common Column Names Within The Tables Should Qualify All References To These Columns.

```
SQL> SELECT  Empno, Ename, Dname, Loc  FROM  Emp, Dept;
SQL> SELECT Empno, Ename, Sal, Grade  FROM  Emp, Salgrade;
SQL> SELECT  Empno, Ename, Dname, Loc, SalGrade  FROM  Emp,  Dept,
SalGrade;
SQL> SELECT Empno, Ename, Dept, Deptno, Dname, Loc  FROM  Emp, Dept;
```

### Join Condition:

- Many Join Queries Contain WHERE Clause, Which Compares Two Columns, Each From a Different Table.
- The Applied Condition is Called a Join CONDITION.
- To Execute a Join…
  - Oracle Combines Pairs of Rows, Each Containing One Row From Each Table, For Which The JOIN Condition Evaluates to TRUE.
- The Columns in The Join Conditions Need Not Be Part of The SELECT List.
- The WHERE Clause of Join Query Can Also Contain Other Conditions That Refer to Columns of Only One Table.
- To Execute a Join of Three or More Tables
  - Oracle First Joins Two of The Tables Based on The Join Conditions Comparing These Columns And Then Join's The Result To Another Join.



- The Oracle Optimizer Determines The Order in Which ORACLE Should Join The Tables Based on…
  - *Given JOIN Condition(s).*
  - *INDEXES Upon The Tables.*
  - *STATISTICS For The Tables.*

- **The LOB Columns Cannot Be Specified in The WHERE Clause, When The WHERE Clause Contains Any JOINS.**

**Syntax:** `WHERE Table1.Column 1 = Table 2.Column 2`

**Guidelines:**

- When Writing a SELECT Statement That JOIN's Tables, Precede The Column Name With The Table Name For Clarity And Enhance Database ACCESS.
- If The Same Column Appears in More Than One Table, The Column Name Must Be Prefixed With The Table Name.
- To Join 'n' Tables Together, We Need a Minimum of 'n-1' Join Conditions.
- The Above Rule Does Not Apply, If The Table Contains a Concatenated Primary Key.

## Equi Joins OR Simple Joins OR Inner Joins:

- An EQUI JOIN is a Join With a Join Condition Containing An Equality Operator.
- It Combines Rows That Have Equivalent Values For The Specified Columns.
- The Total Size of Columns in The Equi Join Condition in a Single Table May Be Limited To The Size of a Data Block minus Some Overhead.
- The Size of The Data Block is Specified By The Initialization Parameter DB_BLOCk_SIZE.

## Qualifying Ambiguous Column Names:

- The Names of The Column Names Should Be Qualified in The WHERE Clause, With The Table Name To Avoid Ambiguity.
- If There Are No Common Column Names Between The Two Tables, The Qualification is Not Necessary But it is Better.

```
SQL> SELECT Emp.Empno Empno, Emp.Ename Ename, Emp.Deptno Deptno,
Dept.Deptno Deptno, Dept.Dname Dname, Dept.Loc Loc FROM Emp, Dept
WHERE Emp.Deptno=Dept.Deptno;

SQL>SELECT Empno, Ename, Emp.Deptno, Loc FROM Emp, Dept WHERE
Emp.Deptno=Dept.Deptno AND Job=UPPER('manager');

SQL>SELECT Empno, Ename, Sal* 12 AnnSal, Emp.Deptno, Loc FROM Emp, Dept
WHERE Emp.Deptno=Dept.Deptno;
```

## Using Table Aliases:

- Table Aliases Can Be Used Instead of Original Table Names.
- A Table Alias Gives an Alternate Name For The Existing Queried Table.
- Table Aliases Help in Keeping The SQL Code Smaller, Hence Using Less Memory.
- The Table Alias is Specified in The FROM Clause.

**Guidelines:**

- A Table Alias Can Be Up To 30 Characters in Length.
- If a Table Alias is Used For a Particular Table Name in The FROM Clause, Then That Table Alias Must Be Substituted For The Table Name Through Out The SELECT Statement.
- A Table Alias Should Be Meaningful and Should Be Maintained as Short as Possible.
- A Table Alias is Valid Only For The Current SELECT Statement Only.

```
SQL> SELECT E.Empno, E.Ename, D.Deptno, D.Dname FROM Emp E, Dept D
WHERE E.Deptno=D.Deptno;

SQL> SELECT E.Ename, E.Job, D.Deptno, D.Dname, D.Loc FROM Emp E, Dept
D WHERE E.Deptno=D.Deptno AND E.Job IN('ANALYST', 'MANAGER');

SQL>SELECT E.Ename, E.Job, D.Dname, D.Loc FROM Emp E, Dept D WHERE
E.Deptno=D.Deptno AND D.Dname <> 'SALES';
```

## Self Joins:

- It is a Join of a Table To Itself.
- The Same Table Appears Twice in The FROM Clause And is Followed By Table Aliases.
- The Table Aliases Must Qualify The Column Names in The Join Condition.
- To Perform a Self Join, Oracle Combines And Returns Rows of The Table That Satisfy The Join Condition.

**Syntax:**

```
SQL> SELECT Columns FROM Table T1, Table T2 WHERE T1.Column 1 = T2.
Column 2
```

**Illustrations:**

```
SQL> SELECT E1. Ename "Employee Name", E2.Ename "Managers Name" FROM
Emp E1, Emp E2 WHERE E1.Mgr =E2.Empno;
SQL>SELECT E1.Ename||'''s Managers is '||E2.Ename "Employees And
Managers" FROM Emp E1, Emp E2 WHERE E1.Mgr = E2. Empno;
SQL> SELECT E1.Ename||' Works For '||E2. Ename "Employees And
Managers" FROM Emp E1, Emp E2 WHERE (E1.Mgr = E2. Empno) AND
E1.Job='CLERK';
```

## Cartesian Products:

- The CARTESIAN PRODUCT is a Join Query, That Does Not Contains a Join Condition.
- During Cartesian Product Oracle Combines Each Row of One Table With Each Row of The Other.
- It Tends to Generate a Large Number of Rows And The Result is Rarely Useful.

```
SQL> SELECT  Ename, Job, Dname  FROM  Emp, Dept;
SQL> SELECT  Ename, Job, Dname  FROM  Emp, Dept  WHERE   Job='MANAGER';
```

## Non Equi Joins:

- It is a Join Condition That is Executed When no Column in One Table Corresponds Directly To a Column in The Other Table.
- The Data in The Tables in Directly Not Related But Indirectly or Logically Related Through Proper Values.

```
SQL>  SELECT  E.Ename, E.Sal, S.Grade  FROM  Emp  E, SalGrade  S  WHERE
E.Sal  BETWEEN  S.Losal  AND  S.Hisal;
SQL>  SELECT  E.Ename, E.Sal, S.Grade  FROM  Emp  E, SalGrade  S  WHERE
(E.Sal  >=  S.Losal  AND  E.Sal <=  S.Hisal)  AND  S.Grade = 1;
```

## Outer Joins:

- An Outer Join Extends The Result of a Simple OR Inner Join.
- An OUTER Join Returns All Rows That Satisfy The Join Condition And Also Those Rows From One Table For Which No Rows From The Other Table Satisfy the Join Condition.
- To Perform An OUTER Join of Tables 'A' and 'B' and Returns All Rows From 'A', Apply The Outer Join Operator '(+)' to All Columns of Table 'B'.
- For all Rows in 'A' That Have no Matching Rows in 'B', Oracle Returns NULL For Any Select List Expressions Containing Columns of 'B'.

**Syntax:**

```
SQL> SELECT  Table 1. Column,  Table 2. Column  FROM  Table 1, Table 2
WHERE  Table 1. Column(+) =Table 2.Column;
SQL> SELECT  Table 1. Column,  Table 2. Column  FROM  Table 1, Table 2
WHERE  Table 1. Column =Table 2.Column (+);
```

**Rules And Restrictions:**

- The (+) Operators Can Appear Only in The WHERE Clause.
- The (+) Operator Can Appear in The Context of The Left Correlation in The FROM Clause , and if 'A' and 'B' Are joined by Multiple Join Condition, We Must Use The (+) Operator in All of These Conditions.
- The (+) Operator Can Be Applied Only to a Column, Not To An Arbitrary Expressions.
- A Condition Cannot Use The IN Comparision Operator To Compare a Column marked With The (+) Operator With an Expression.
- A Condition Cannot Compare Any Column Marked With The (+) Operator With a Sub Query.

```
SQL> SELECT E.Ename, D.Deptno, D.Dname FROM Emp E, Dept D WHERE
E.Deptno(+)= D.Deptno ORDER BY E.Deptno;
SQL> SELECT E.Ename, D.Deptno, D.Dname FROM Emp E, Dept D WHERE
E.Deptno(+)= D.Deptno AND E.Deptno(+)=10 ORDER BY E.Deptno;
SQL> SELECT E.Ename, D.Deptno, D.Dname FROM Emp E, Dept D WHERE
E.Deptno= D.Deptno(+) AND E.Deptno(+)=10 ORDER BY E.Deptno;
SQL> SELECT E.Ename Employee, NVL(M.Ename, 'Supreme Authority') Manager
FROM Emp E, Emp M WHERE E.MGR = M.Empno(+);
```

**Joining Data From More Than Two Tables:**

- Joins Can Be Established on More Than Two Tables.
- The Join is First Executed Upon The Two Most Relevant Tables And Then The Result is Applied Upon the Third Table.

```
SQL> SELECT E.Ename, E.Deptno, M.Ename Manger, M.Deptno From Emp E,
Dept D, Emp M WHERE E.MGR =M.Empno AND E.Deptno = D.Deptno;
```

**Categories of Joins:**

- **Oracle Proprietary Joins (8i And Prior)**
  - Equi Join
  - Non-Equi Join
  - Outer Join
  - Self Join
- **ANSI SQL: 1999 Compliant Joins:**
  - Cross Joins
  - Natural Joins
  - Using Clause
  - Full OR Two Sided Outer Joins
  - Arbitrary Join Conditions For Outer Joins

- **ISO OR ANSI Joins:**

  - **Cross Join:**

```
SQL> SELECT Ename, Deptno, Dname,Loc FROM Emp CROSS JOIN Dept WHERE
Emp.Deptno=Dept.Deptno;
```

  - **Natural Join:**

```
SQL> SELECT Ename, Deptno, Dname, Loc FROM Emp NATURAL JOIN Dept;
```

  - **USING Clause:**

```
SQL> SELECT Ename, Deptno, Dname, Loc FROM Emp JOIN Dept USING(Deptno);
```

  - **INNER JOIN:**

```
SQL> SELECT Ename, Deptno.Deptno, Dname, Loc FROM Emp JOIN Dept ON
Emp.Deptno=Dept.Deptno;
```

  - **Self Join:**

```
SQL> SELECT E.Ename Employee, M.Ename Manager FROM Emp E INNER
JOIN Emp M ON(E.MGR=M.Empno);
```

  - **Join On More Than Two Tables(Cascading Style Of Joins):**

```
SQL> SELECT Ename, Sal, Grade, Dept. Deptno, Dname FROM Emp JOIN Dept
ON Emp.Deptno = Dept.Deptno JOIN SalGrade ON Emp.Sal BETWEEN LoSal
AND HiSal;
SQL> SELECT Ename, M.Ename, Sal, Grade, D.Deptno, Dname FROM Emp E
INNER JOIN Dept D ON E.Deptno =D.Deptno INNER JOIN Emp MON
E.Empno=M.MGR INNER JOIN SalGrade S ON E.Sal BETWEEN LoSal AND
HiSal;
```

  - **Right Outer Join:**

```
SQL> SELECT Ename, Dept.Deptno, Dname, Loc FROM Emp RIGHT JOIN Dept
ON Emp.Deptno=Dept.Deptno;
```

  - **Left Outer Join:**

```
SQL> SELECT Ename, Dept.Deptno, Dname, Loc FROM Emp LEFT JOIN Dept
ON Emp.Deptno=Dept.Deptno;
```

  - **FULL Join:**

```
SQL>SELECT Ename, Dept.Deptno, Dname, Loc FROM Emp FULL JOIN Dept ON
Dept.Deptno=Emp.Deptno;
SQL>SELECT Ename, Dept.Deptno, Dname, Loc FROM Emp FULL JOIN Dept ON
Emp.Deptno=Dept.Deptno;
```