# Oracle Joins

Join is a query that is used to combine rows from two or more tables, views, or materialized views. It retrieves data from multiple tables and creates a new table.

## Join Conditions

There may be at least one join condition either in the FROM clause or in the WHERE clause for joining two tables. It compares two columns from different tables and combines pair of rows, each containing one row from each table, for which join condition is true.

## Types of Joins

* Inner Joins (Simple Join)
* Outer Joins
  + Left Outer Join (Left Join)
  + Right Outer Join (Right Join)
  + Full Outer Join (Full Join)
* Equijoins
* Self Joins
* Cross Joins (Cartesian Products)
* Antijoins
* Semijoins

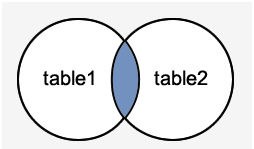
# Oracle INNER JOIN

Inner Join is the simplest and most common type of join. It is also known as simple join. It returns all rows from multiple tables where the join condition is met.

Syntax

1. **SELECT** columns
2. **FROM** table1
3. **INNER** JOIN table2
4. **ON** table1.**column** = table2.**column**;

Image representation of Inner Join

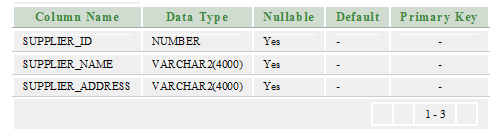
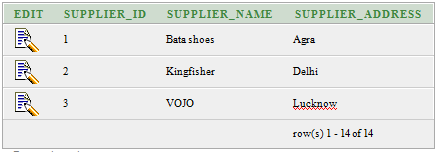


1. **SELECT** suppliers.supplier\_id, suppliers.supplier\_name, order1.order\_number
2. **FROM** suppliers
3. **INNER** JOIN order1
4. **ON** suppliers.supplier\_id = order1.supplier\_id;

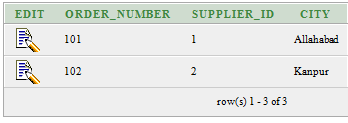
## Oracle INNER JOIN Example

Let's take an example to perform Inner Join on two tables "Suppliers" and "Order1".

Suppliers

Order1

# Oracle OUTER JOIN

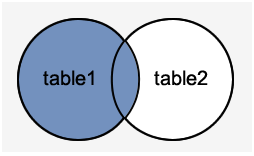
An outer join is similar to equijoin but it gets also the non-matched rows from the table. It is categorized in Left Outer Join, Right Outer Join and Full Outer Join by Oracle 9i ANSI/ISO 1999 standard.

## Left Outer Join

Left Outer Join returns all rows from the left (first) table specified in the ON condition and only those rows from the right (second) table where the join condition is met.

Syntax

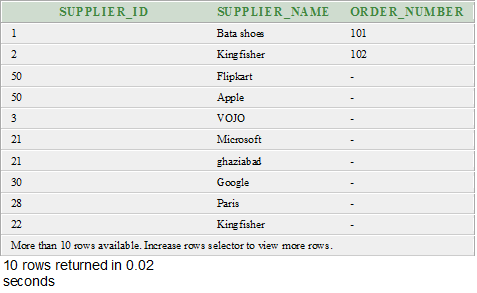
1. **SELECT** columns
2. **FROM** table1
3. LEFT [OUTER] JOIN table2
4. **ON** table1.**column** = table2.**column**;



Execute this query

1. **SELECT** suppliers.supplier\_id, suppliers.supplier\_name, order1.order\_number
2. **FROM** suppliers
3. LEFT OUTER JOIN order1
4. **ON** suppliers.supplier\_id = order1.supplier\_id;

Output

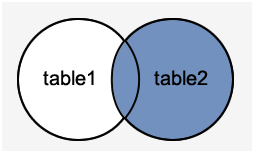


## Right Outer Join

The Right Outer Join returns all rows from the right-hand table specified in the ON condition and only those rows from the other table where the join condition is met.

Syntax

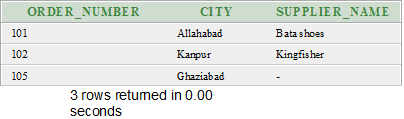
1. **SELECT** columns
2. **FROM** table1
3. RIGHT [OUTER] JOIN table2
4. **ON** table1.**column** = table2.**column**;



**SELECT** order1.order\_number, order1.city, suppliers.supplier\_name

1. **FROM** suppliers
2. RIGHT OUTER JOIN order1
3. **ON** suppliers.supplier\_id = order1.supplier\_id;

Output

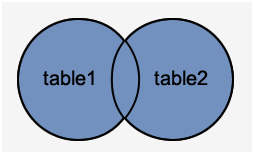


## Full Outer Join

The Full Outer Join returns all rows from the left hand table and right hand table. It places NULL where the join condition is not met.

Syntax

1. **SELECT** columns
2. **FROM** table1
3. **FULL** [OUTER] JOIN table2
4. **ON** table1.**column** = table2.**column**;



1. **SELECT** suppliers.supplier\_id, suppliers.supplier\_name, order1.order\_number
2. **FROM** suppliers
3. **FULL** OUTER JOIN order1
4. **ON** suppliers.supplier\_id = order1.supplier\_id;

Output

# Oracle EQUI JOIN

Oracle Equi join returns the matching column values of the associated tables. It uses a comparison operator in the WHERE clause to refer equality.

Syntax

1. **SELECT** column\_list
2. **FROM** table1, table2....
3. **WHERE** table1.column\_name =
4. table2.column\_name;

Equijoin also can be performed by using JOIN keyword followed by ON keyword and then specifying names of the columns along with their associated tables to check equality.

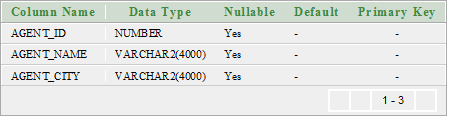
Syntax

1. **SELECT** \*
2. **FROM** table1
3. JOIN table2
4. [**ON** (join\_condition)]

## Oracle EQUI JOIN Example

Let' take two tables "agents" and "customer".

Agents table



Agent data



# Oracle SELF JOIN

Self Join is a specific type of Join. In Self Join, a table is joined with itself (Unary relationship). A self join simply specifies that each rows of a table is combined with itself and every other row of the table.

Syntax

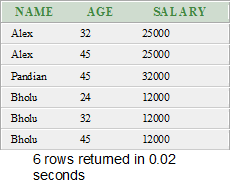
1. **SELECT** a.column\_name, b.column\_name...
2. **FROM** table1 a, table1 b
3. **WHERE** a.common\_filed = b.common\_field;

## Oracle SELF JOIN Example

Let's take a table "customers".



1. **SELECT**  a.**name**, b.age, a.SALARY
2. **FROM** CUSTOMERS a, CUSTOMERS b
3. **WHERE** a.SALARY < b.SALARY;



# Oracle Cross Join (Cartesian Products)

The CROSS JOIN specifies that all rows from first table join with all of the rows of second table. If there are "x" rows in table1 and "y" rows in table2 then the cross join result set have x\*y rows. It normally happens when no matching join columns are specified.

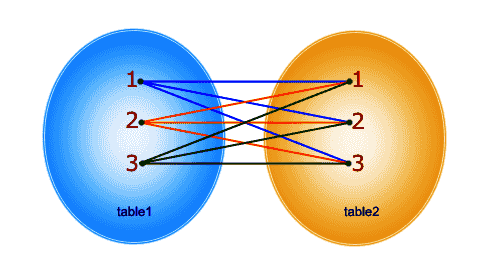
In simple words you can say that if two tables in a join query have no join condition, then the Oracle returns their Cartesian product.

Syntax

1. **SELECT** \*
2. **FROM** table1
3. CROSS JOIN table2;

Or

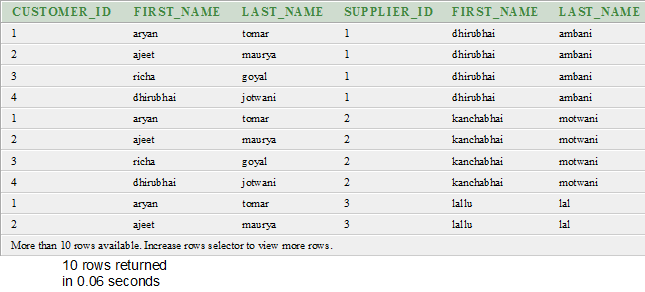
1. **SELECT** \* **FROM** table1, table2



1. **CREATE** **TABLE**  "CUSTOMER"
2. (    "CUSTOMER\_ID" NUMBER,
3. "FIRST\_NAME" VARCHAR2(4000),
4. "LAST\_NAME" VARCHAR2(4000)
5. )
6. **CREATE** **TABLE**  "SUPPLIER"
7. (    "SUPPLIER\_ID" NUMBER,
8. "FIRST\_NAME" VARCHAR2(4000),
9. "LAST\_NAME" VARCHAR2(4000)
10. )

Execute this query

1. **SELECT** \* **FROM** customer,supplier



# Oracle Anti Join

Anti-join is used to make the queries run faster. It is a very powerful SQL construct Oracle offers for faster queries.

Anti-join between two tables returns rows from the first table where no matches are found in the second table. It is opposite of a semi-join. An anti-join returns one copy of each row in the first table for which no match is found.

Anti-joins are written using the NOT EXISTS or NOT IN constructs.

**CREATE** **TABLE**  "DEPARTMENTS"

1. (    "DEPARTMENT\_ID" NUMBER(10,0) NOT NULL ENABLE,
2. "DEPARTMENT\_NAME" VARCHAR2(50) NOT NULL ENABLE,
3. **CONSTRAINT** "DEPARTMENTS\_PK" **PRIMARY** **KEY** ("DEPARTMENT\_ID") ENABLE
4. )
5. /

**CREATE** **TABLE**  "CUSTOMER"

1. (    "CUSTOMER\_ID" NUMBER,
2. "FIRST\_NAME" VARCHAR2(4000),
3. "LAST\_NAME" VARCHAR2(4000),
4. "DEPARTMENT\_ID" NUMBER
5. )

**SELECT**   departments.department\_id, departments.department\_name

1. **FROM**     departments
2. **WHERE**    NOT EXISTS
3. (
4. **SELECT** 1
5. **FROM**   customer
6. **WHERE** customer.department\_id = departments.department\_id
7. )
8. **ORDER** **BY** departments.department\_id;

# Oracle Semi Join

Semi-join is introduced in Oracle 8.0. It provides an efficient method of performing a WHERE EXISTS sub-query.

A semi-join returns one copy of each row in first table for which at least one match is found.

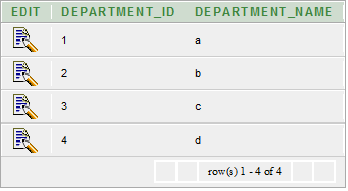
Semi-joins are written using the EXISTS construct.

## Oracle Semi Join Example

Let's take two tables "departments" and "customer"

Departments table

1. **CREATE** **TABLE**  "DEPARTMENTS"
2. (    "DEPARTMENT\_ID" NUMBER(10,0) NOT NULL ENABLE,
3. "DEPARTMENT\_NAME" VARCHAR2(50) NOT NULL ENABLE,
4. **CONSTRAINT** "DEPARTMENTS\_PK" **PRIMARY** **KEY** ("DEPARTMENT\_ID") ENABLE
5. )
6. /



Customer table

1. **CREATE** **TABLE**  "CUSTOMER"
2. (    "CUSTOMER\_ID" NUMBER,
3. "FIRST\_NAME" VARCHAR2(4000),
4. "LAST\_NAME" VARCHAR2(4000),
5. "DEPARTMENT\_ID" NUMBER
6. )
7. /



Execute this query

1. **SELECT**   departments.department\_id, departments.department\_name
2. **FROM**     departments
3. **WHERE**    EXISTS
4. (
5. **SELECT** 1
6. **FROM**   customer
7. **WHERE** customer.department\_id = departments.department\_id
8. )
9. **ORDER** **BY** departments.department\_id;

Output

