Oracle Inner JOIN

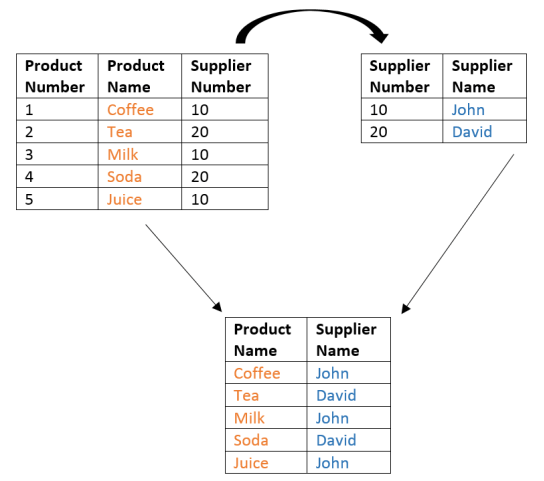
This [SQL tutorial](http://ramkedem.com/en/sql-tutorial/) focuses on the Oracle **Inner Join statement**, and provides explanations, examples and exercises. For this lesson’s exercises, use this [link](http://ramkedem.com/en/practice-sql-join-methods/).

This tutorial is a part of several posts describing how to use the JOIN statement in Oracle. To read additional posts regarding this subject, please use the following links:

* Oracle Inner JOIN – Joining data items from tables, based on values common to both tables.
* [Oracle Outer JOIN](http://ramkedem.com/en/oracle-outer-join/)– Joining data items from tables, based on values common to both tables, while displaying all data from one table regardless of if there is a match on the second table.
* [Oracle Self JOIN](http://ramkedem.com/en/oracle-self-join/)– Join a table to itself.

Oracle Inner JOIN

Sometimes you may need to use data from more than one table. In this illustration, the result set displays information from two separate tables: product name from Products table, supplier name from Suppliers table.

[](https://i0.wp.com/ramkedem.com/wp-content/uploads/2015/08/oracle_querying_two_tables.png)

* Multiple tables (two or more tables) can be linked only if they have common values (in this case, supplier number) or a logical connection of some kind.
* Relating between two tables requires you to determine the join condition. In the example shown above, the join condition was based on the equality operator (=).

Joining Two Tables

|  |  |
| --- | --- |
|  | SELECT table\_a.column\_name , table\_b.column\_name, table\_a.column\_name ..  FROM   table\_a , table\_b  WHERE  table\_a.column\_name = table\_b.column\_name |

For example, we would like to retrieve data from two tables – Products and Suppliers. We would like to retrieve the product’s name and price from the Products table and the supplier’s name from the Suppliers table:

|  |  |
| --- | --- |
|  | SELECT products.product\_name , products.price,  suppliers.supplier\_name  FROM   products , suppliers  WHERE  products.supplier\_id = suppliers.supplier\_id |

* In the Oracle SELECT clause, precede the column name with the table name for clarity.
* When a column is common to both tables, it must be prefixed with the table name.
* In the Oracle FROM clause, you need to specify the tables from which you would like to retrieve the data. These tables are specified with comma (,) between them.
* After the WHERE keyword, specify the join condition.
* To determine the relation between Products and Suppliers tables – values in the supplier\_id column on both tables must be equal. This type of relation is referred as an **Equi Join.**
* **Equi joins**are also called **Simple Joins** or **Inner Joins.**
* Frequently, this relation involves primary key and foreign key complements.

Oracle Table Aliases

Qualifying column names with table names can be time consuming, and may result in a very long, unreadable query. In Oracle, Instead of writing a full table name after each column, use Table Aliases. Just as Column Alias gives a column another name, a table alias gives a table another name.

Let us repeat the task shown in the previous example, this time by using Table Aliases:

|  |  |
| --- | --- |
|  | SELECT prd.product\_name , prd.price , sup.supplier\_name  FROM   products prd ,suppliers sup  WHERE  prd.supplier\_id = sup.supplier\_id |

* Use the Oracle FROM clause to define the table aliases.
* Write an alias after each table name.
* You can assign any alias to a table (for example, you can assign the letter A to the Products table); however, it is advisable to assign meaningful aliases.
* After defining an alias to a table, that alias must be substituted for the table name throughout the Oracle SELECT statement. All explicit references to the table must use the table alias, not the table name.
* The table alias is valid only for the current Oracle SELECT statement.

Adding Additional Clauses

you can carry on adding additional clauses to your Oracle SELECT statement (restricting the rows returned, adding aggregations, defining the sorting order and so on).

Below are several Oracle examples:

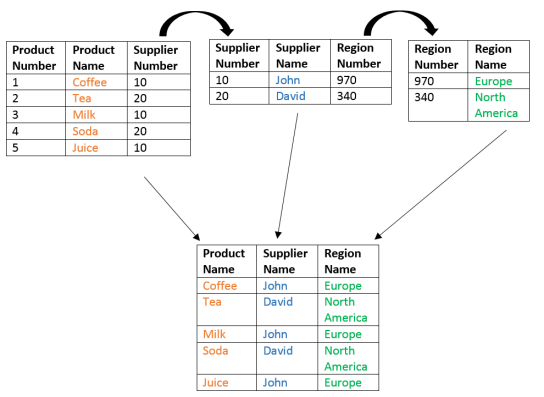
This Oracle example would retrieve all products whose price is greater than 60 and their supplier number equals 90:

|  |  |
| --- | --- |
|  | SELECT prd.product\_name , prd.price , sup.supplier\_name  FROM   products prd , suppliers sup  WHERE  prd.supplier\_id = sup.supplier\_id  AND prd.supplier\_id = 90  AND prd.price > 60 |

You can change the sorting order, thereby displaying the data items sorted by the product’s price:

|  |  |
| --- | --- |
|  | SELECT  prd.product\_name , prd.price , sup.supplier\_name  FROM   products prd JOIN suppliers sup  WHERE  prd.supplier\_id = sup.supplier\_id  AND prd.supplier\_id = 90  AND prd.price > 60  ORDER BY prd.price DESC |

Joining More than Two Tables

[](https://i1.wp.com/ramkedem.com/wp-content/uploads/2015/08/querying_three_tables.png)

Sometimes you may need to join more than two tables, for example: displaying the product’s name from the Product table, the name of the supplier from Suppliers table, and the name of the region where this supplier resides from the Regions table.

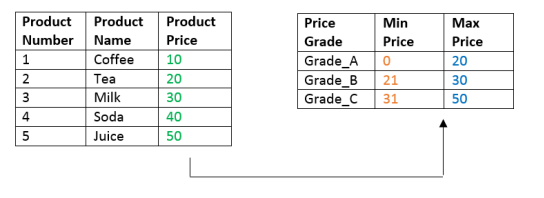
Joining an additional table requires us to:

* Specify the table name in the Oracle FROM clause (by separating one table from another by comma).
* Determine the additional join condition, and specify additional join condition in the Oracle WHERE clause.

|  |  |
| --- | --- |
|  | SELECT prd.product\_name , prd.price , sup.supplier\_name, reg.region\_name  FROM  products prd , suppliers sup , regions reg  WHERE prd.supplier\_id = sup.supplier\_id  AND  sup.region\_id = reg.region\_id |

* The same concept applies to joining four tables or more – adding the table name at the Oracle FROM clause, and specifying additional join condition at the Oracle WHERE clause.

None Equi Join

A Oracle None Equi Join is a join condition containing something other than equality operator. For example: sometimes you would like to join two tables that do not have a shared column, and seemingly have no join condition:[](https://i0.wp.com/ramkedem.com/wp-content/uploads/2015/08/oracle_none_equi_join.png)

What is product number 4 (Soda) price grade? in this case there is no shared column, but you can tell that it belongs to price grade: Grade\_C , because when comparing its price to the data found in the Price Grades table, it turned out that its price is between Min price (31) and Max price (50) of Grade\_C.

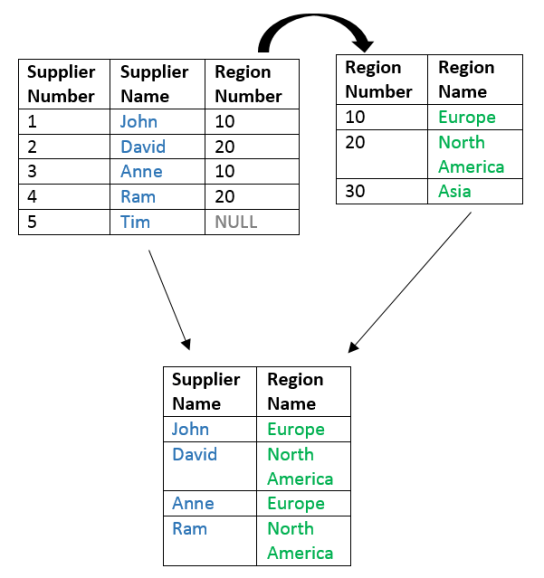
This relation is obtained using the Oracle BETWEEN operator :

|  |  |
| --- | --- |
|  | SELECT prd.product\_name , prd.price , pg.grade\_level  FROM   products prd , price\_grades pg  WHERE prd.price BETWEEN pg.lowest\_sal AND pg.highest\_sal |

You can join multiple tables (two or more) as long as you can find any logical condition that can be used for joining these tables.

OUTER JOIN

While the Oracle Inner JOIN statement allows us to retrieve rows from both tables only if they match the join condition, Oracle Outer JOIN statement retrieves all rows from at least one of the tables, regardless of whether there is a match on the second table.

[](https://i2.wp.com/ramkedem.com/wp-content/uploads/2015/08/oracle_inner_join.png)

The illustration above shows the query result of using the Oracle Inner JOIN in order to join Suppliers and Regions tables.

Note that Tim does not appear, and neither does Asia region. The reason for this is the way those tables were compared. The comparison between these two tables was performed by using the following condition:

|  |  |
| --- | --- |
| 1 | WHERE sup.region\_id = reg.region\_id |

That means that as long as the values of the column Region Number in the Suppliers table are equal to the values of the column Region Number in the Regions table, the row will appear in the query result. If a row does not satisfy a join condition, the row will not appear in the query result.

* Tim does not appear because his region’s number is NULL, and NULL cannot be compared to any value.
* The Asia region does not appear because its number is 30, and this value cannot be compared to any value in the column Region Number within the Suppliers table.

To display all of the data items that are found on one table (either left or right), including values that have no comparable data on the second table, use either Oracle Left Outer Join or Oracle Right Outer Join.

Oracle Left OUTER JOIN

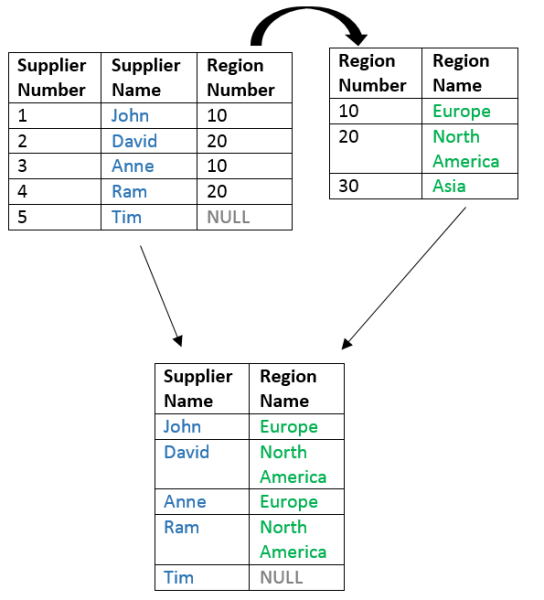
If the Suppliers table, as described in the illustration above, appears on the left side of the Oracle JOIN condition.

|  |  |
| --- | --- |
|  | WHERE  sup.supplier\_id = reg.supplier\_id  (left)              (right) |

To display all of the Suppliers who exist on the table, including those who do not belong to any region, use the Oracle Outer Join operator. This operator, a plus sign enclosed withing parentheses (+), will be placed on the side of the join condition that is deficient of information – the right side.

|  |  |
| --- | --- |
|  | SELECT   sup.last\_name , sup.salary , reg.region\_name  FROM    Suppliers sup, Regions reg  WHERE   sup.region\_id = reg.region\_id (+) |

Joining the tables by using Left Outer Join results in displaying all of the Suppliers, including the Suppliers who do not belong to any region (5 rows in total):

[](https://i2.wp.com/ramkedem.com/wp-content/uploads/2015/08/oracle_left_outer_join.png)

Right OUTER JOIN

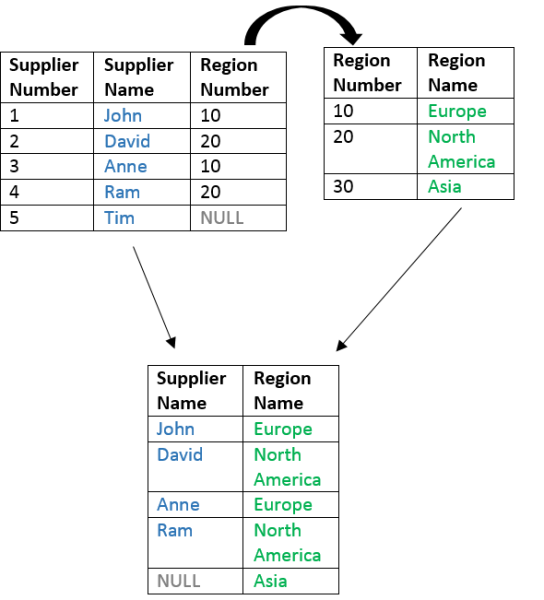
If the Regions table, as described in the illustration above, appears on the right side of the Oracle JOIN statement.

|  |  |
| --- | --- |
|  | WHERE sup.region\_id = reg.region\_id  (left)             (right) |

To display all of the Regions that exist on the table, including those without any Suppliers, use the Oracle Outer Join operator. This operator, a plus sign enclosed withing parentheses (+), will be placed on the side of the join condition that is deficient of information – the left side.

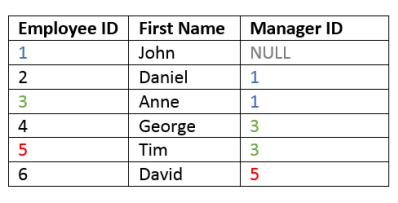
|  |  |
| --- | --- |
|  | SELECT   sup.last\_name , sup.salary , reg.region\_name  FROM     Suppliers sup, Regions reg  WHERE    sup.region\_id (+) = reg.region\_id |

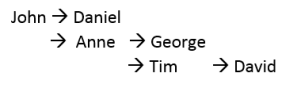
Joining the tables by using the Right Outer JOIN results in displaying all of the Regions, including the Regions without any Suppliers (5 rows in total):

[](https://i1.wp.com/ramkedem.com/wp-content/uploads/2015/08/right_outer_join.png)

## Oracle Self JOIN

The Oracle Self Join allows joining a table to itself. To fully understand this concept, please consider the following example: the table in the illustration below contains information about employees and their managers. If you’ll take a closer look, you’ll notice this table has a certain hierarchy of employees and managers, where a certain employee may also be a manager (for example employee number 5 – Tim, is also the manager of David).

[](https://i1.wp.com/ramkedem.com/wp-content/uploads/2014/12/sql_self_join.png)

[](https://i1.wp.com/ramkedem.com/wp-content/uploads/2014/12/sql_self_join_hierarchy.png)

Since the table references data in itself, you can answer questions such as: what is the name of David’s manager? To find out, check what is the manager ID of David’s manager (5), and then check to which employee this number corresponds (Tim).

Employee’s manager ID = “Manager’s” employee ID.

On tables of this type you may sometimes want to display data that relies on this relation. You can use this relation to create a query that retrieves the employee’s name and the name of this employee’s manager.

|  |  |
| --- | --- |
|  | SELECT emp.employee\_id , emp.last\_name , emp.manager\_id ,  mng.last\_name  FROM   employees emp ,employees mng  WHERE  emp.manager\_id = mng.employee\_id |

Because this operation joins the same table to itself, it actually involves referring the same table twice, each time under a different alias (once as “Employees” table, once as “Managers”).

The Oracle WHERE clause is used for defining the relation of the table to itself. In this case, the number of the employee’s manager (“Emp.Manager\_ID”) is compared with the employee number of this manager (“Mng.Employee\_ID”).