ORACLETUTORIAL.COM

Oracle Joins

Summary: in this tutorial, you will learn various kind of Oracle joins that allow you to query data from two or more related tables.

Oracle join is used to combine columns from two or more tables based on values of the related columns. The related columns are typically the primary key (https://www.oracletutorial.com/oracle-basics/oracle-primary-key/) column(s) of the first table and foreign key (https://www.oracletutorial.com/oracle-basics/oracle-foreign-key/) column(s) of the second table.

Oracle supports inner join (https://www.oracletutorial.com/oracle-basics/oracle-inner-join/) , left join (https://www.oracletutorial.com/oracle-basics/oracle-left-join/) , right join (https://www.oracletutorial.com/oracle-basics/oracle-right-join/) , full outer join (https://www.oracletutorial.com/oracle-basics/oracle-full-outer-join/) and cross join (https://www.oracletutorial.com/oracle-basics/oracle-cross-join/) .

Note that you can join a table to itself to query hierarchical data using an inner join, left join, or right join. This kind of join is known as self-join (https://www.oracletutorial.com/oracle-basics/oracle-self-join/) .

Setting up sample tables

We will create two new tables (https://www.oracletutorial.com/oracle-basics/oracle-create-table/) with the same structure for the demonstration:

```
CREATE TABLE palette_a (
    id INT PRIMARY KEY,
    color VARCHAR2 (100) NOT NULL
);

CREATE TABLE palette_b (
    id INT PRIMARY KEY,
    color VARCHAR2 (100) NOT NULL
);
```

```
INSERT INTO palette_a (id, color)
VALUES (1, 'Red');

INSERT INTO palette_a (id, color)
VALUES (2, 'Green');

INSERT INTO palette_a (id, color)
VALUES (3, 'Blue');

INSERT INTO palette_a (id, color)
VALUES (4, 'Purple');

-- insert data for the palette_b
INSERT INTO palette_b (id, color)
VALUES (1, 'Green');

INSERT INTO palette_b (id, color)
VALUES (2, 'Red');

INSERT INTO palette_b (id, color)
VALUES (3, 'Cyan');

INSERT INTO palette_b (id, color)
VALUES (4, 'Brown');
```

The tables have some common colors such as `Red` and `Green`. Let's call the `palette_a` the left table and `palette_b` the right table:

ID	COLOR
1	Red
2	Green
3	Blue
4	Purple

ID	COLOR
1	Green
2	Red
3	Cyan
4	Brown

Oracle inner join

The following statement joins the left table to the right table using the values in the `color` column:

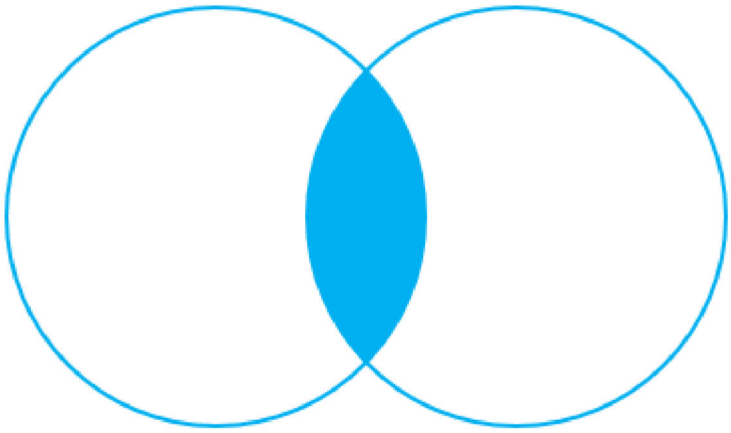
```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
INNER JOIN palette_b b ON a.color = b.color;
```

Here is the output:

ID_A	COLOR_A	ID_B	COLOR_B
2	Green	1	Green
1	Red	2	Red

As can be seen clearly from the result, the inner join returns rows from the left table that match with the rows from the right table.

The following Venn diagram illustrates an inner join when combining two result sets:



INNER JOIN

Oracle left join

The following statement joins the left table with the right table using a left join (or a left outer join):

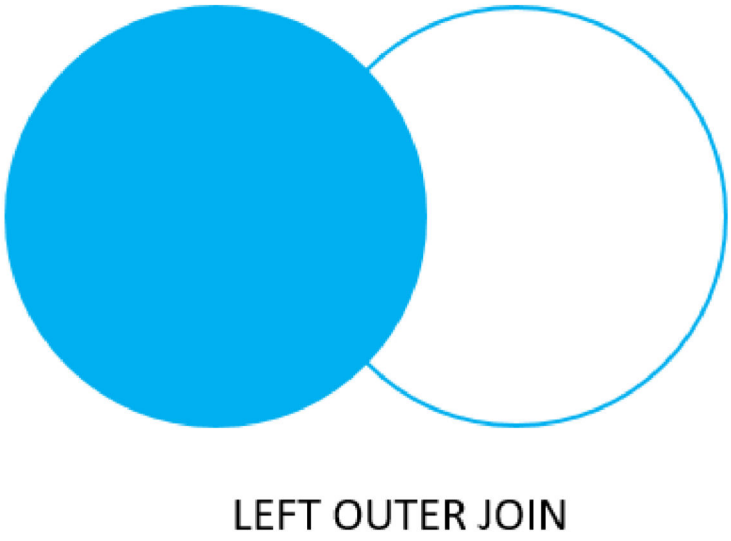
```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
LEFT JOIN palette_b b ON a.color = b.color;
```

The output is shown as follows:

ID_A	COLOR_A	ID_B	COLOR_B
2	Green	1	Green
1	Red	2	Red
3	Blue	(null)	(null)
4	Purple	(null)	(null)

The left join returns all rows from the left table with the matching rows if available from the right table. If there is no matching row found from the right table, the left join will have null values for the columns of the right table:

The following Venn diagram illustrates the left join:



Sometimes, you want to get only rows from the left table that do not exist in the right table. To achieve this, you use the left join and a `WHERE` clause to exclude the rows from the right table.

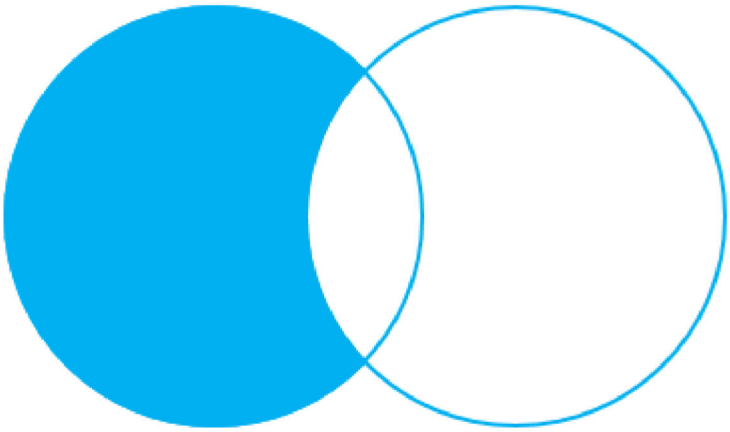
For example, the following statement shows colors that only available in the `palette_a` but not `palette_b` :

```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
LEFT JOIN palette_b b ON a.color = b.color
WHERE b.id IS NULL;
```

Here is the output:

ID_A	COLOR_A	ID_B	COLOR_B
3	Blue	(null)	(null)
4	Purple	(null)	(null)

The following Venn diagram illustrates the left join with the exclusion of rows from the right table:



LEFT OUTER JOIN – only rows from the left table

Oracle right join

The right join or right outer join is a reversed version of the left join. The right join makes a result set that contains all rows from the right table with the matching rows from the left table. If there is no match, the left side will have nulls.

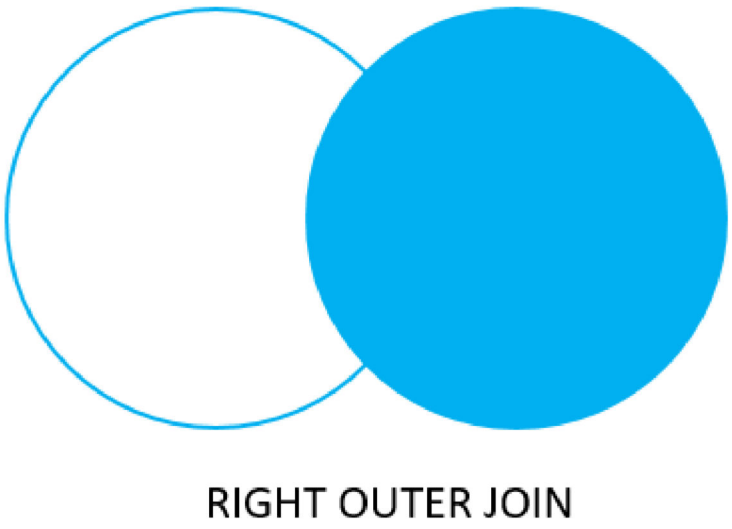
The following example use right join to join the left table to the right table:

```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
RIGHT JOIN palette_b b ON a.color = b.color;
```

Here is the output:

ID_A	COLOR_A	ID_B	COLOR_B
1	Red	2	Red
2	Green	1	Green
(null)	(null)	4	Brown
(null)	(null)	3	Cyan

The following Venn diagram illustrates the right join:



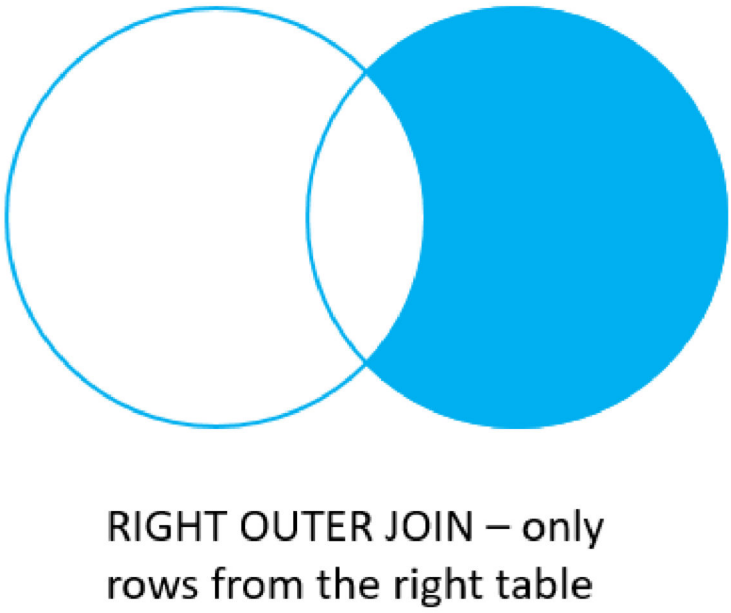
Likewise, you can get only rows from the right table but not the left table by adding a WHERE clause to the above statement as shown in the following query:

```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
RIGHT JOIN palette_b b ON a.color = b.color
WHERE a.id IS NULL;
```

Here is the output:

ID_A	COLOR_A	ID_B	COLOR_B
(null)	(null)	4	Brown
(null)	(null)	3	Cyan

The following Venn diagram illustrates the right join with the exclusion of rows from the left table:



Oracle full outer join

Oracle full outer join (https://www.oracletutorial.com/oracle-basics/oracle-full-outer-join/) or full join (https://www.oracletutorial.com/oracle-basics/oracle-full-outer-join/) returns a result set that contains all rows from both left and right tables, with the matching rows from both sides where available. If there is no match, the missing side will have nulls.

The following example shows the full outer join of the left and right tables:

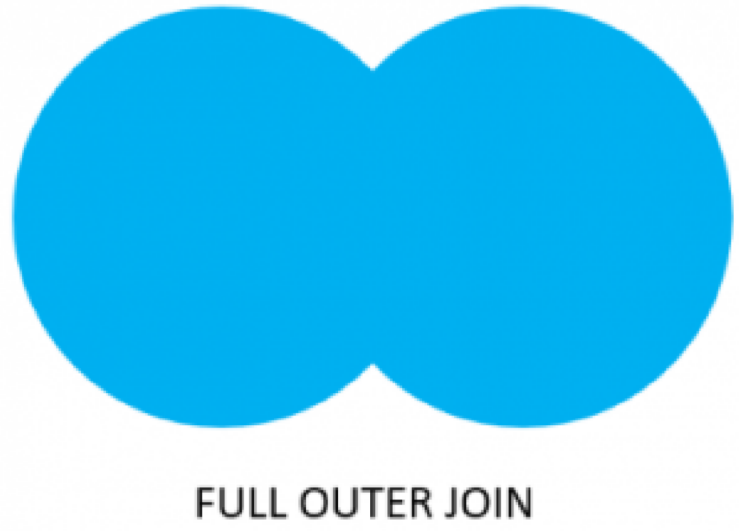
```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
FULL OUTER JOIN palette_b b ON a.color = b.color;
```

The following picture illustrates the result set of the full outer join:

ID_A	COLOR_A	ID_B	COLOR_B
1	Red	2	Red
2	Green	1	Green
3	Blue	(null)	(null)
4	Purple	(null)	(null)
(null)	(null)	4	Brown
(null)	(null)	3	Cyan

Note that the `OUTER` keyword is optional.

The following Venn diagram illustrates the full outer join:



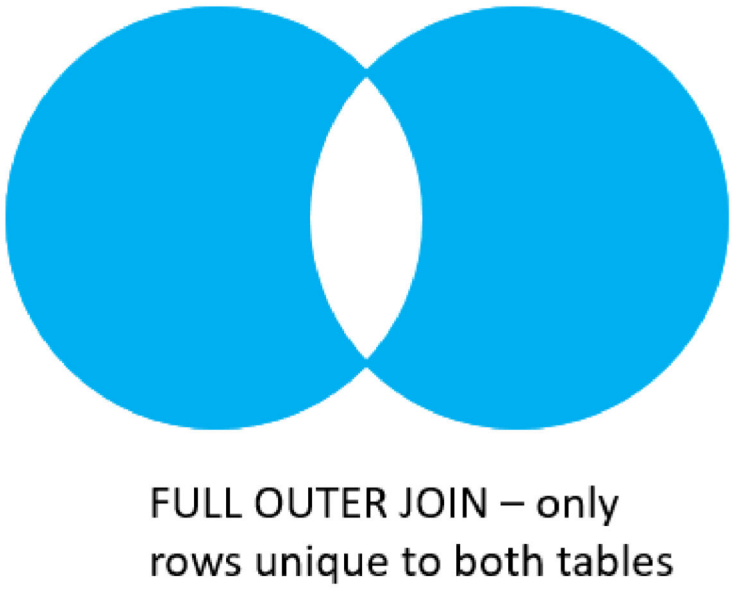
To get a set of rows that are unique from the left and right tales, you perform the same full join and then exclude the rows that you don't want from both sides using a `WHERE` (https://www.oracletutorial.com/oracle-basics/oracle-where/) clause as follows:

```
SELECT
    a.id id_a,
    a.color color_a,
    b.id id_b,
    b.color color_b
FROM
    palette_a a
FULL JOIN palette_b b ON a.color = b.color
WHERE a.id IS NULL OR b.id IS NULL;
```

Here is the result:

ID_A	COLOR_A	ID_B	COLOR_B
(null)	(null)	3	Cyan
(null)	(null)	4	Brown
3	Blue	(null)	(null)
4	Purple	(null)	(null)

The following Venn diagram illustrates the above operation:



In this tutorial, you have learned how to use various kinds of Oracle joins to query data from two tables.