PostgreSQL Joins

Summary: in this tutorial, you will learn about various kinds of PostgreSQL joins including inner join, left join, right join, and full outer join.

PostgreSQL join is used to combine columns from one (self-join (https://www.postgresqltutorial.com /postgresql-tutorial/postgresql-self-join/)) or more tables based on the values of the common columns between related tables. The common columns are typically the primary key (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-primary-key/) columns of the first table and foreign key (https://www.postgresqltutorial.com/postgresql-foreign-key/) columns of the second table.

PostgreSQL supports inner join (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-inner-join/) , left join (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-left-join/) , right join (https://www.postgresqltutorial.com/postgresql-right-join/) , full outer join (https://www.postgresqltutorial.com /postgresql-tutorial/postgresql-full-outer-join/) , cross join (https://www.postgresqltutorial.com/postgresql-tutorial /postgresql-cross-join/) , natural join (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-natural-join/) , and a special kind of join called self-join (https://www.postgresqltutorial.com/postgresql-tutorial /postgresql-self-join/) .

Setting up sample tables

Suppose you have two tables called `basket_a` and `basket_b` that store fruits:

```
CREATE TABLE basket_a (
    a INT PRIMARY KEY,
    fruit_a VARCHAR (100) NOT NULL
);


CREATE TABLE basket_b (
    b INT PRIMARY KEY,
```

```
    fruit_b VARCHAR (100) NOT NULL
);


INSERT INTO basket_a (a, fruit_a)
VALUES
    (1, 'Apple'),
    (2, 'Orange'),
    (3, 'Banana'),
    (4, 'Cucumber');


INSERT INTO basket_b (b, fruit_b)
VALUES
    (1, 'Orange'),
    (2, 'Apple'),
    (3, 'Watermelon'),
    (4, 'Pear');
```

The tables have some common fruits such as `apple` and `orange`.

The following statement returns data from the `basket_a` table:

	a integer	fruit_a character varying (100)
1	1	Apple
2	2	Orange
3	3	Banana
4	4	Cucumber

And the following statement returns data from the `basket_b` table:

|	b integer	fruit_b character varying (100)
1	1	Orange
2	2	Apple
3	3	Watermelon
4	4	Pear

PostgreSQL inner join

The following statement joins the first table (basket_a) with the second table (basket_b) by matching the values in the fruit_a and fruit_b columns:

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
INNER JOIN basket_b
    ON fruit_a = fruit_b;
```

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange

The inner join examines each row in the first table (basket_a). It compares the value in the fruit_a column with the value in the fruit_b column of each row in the second table (basket_b). If these values are equal, the inner join creates a new row that contains columns from both tables and adds this new row the result set.

The following Venn diagram illustrates the inner join:
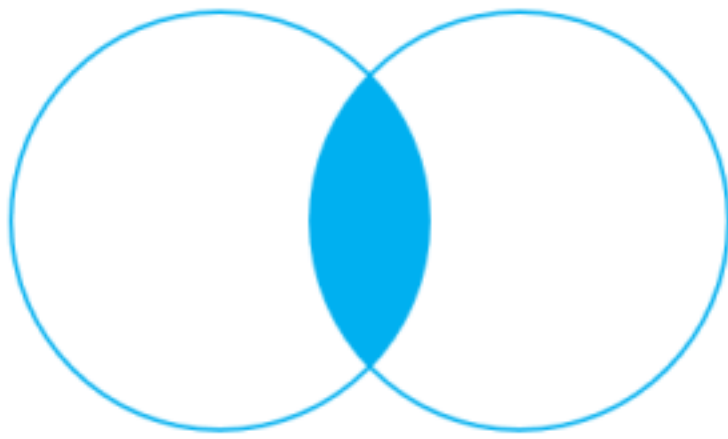
INNER JOIN

PostgreSQL left join

The following statement uses the left join clause to join the `basket_a` table with the `basket_b` table. In the left join context, the first table is called the left table and the second table is called the right table.

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
LEFT JOIN basket_b
    ON fruit_a = fruit_b;
```

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange
3	3	Banana	[null]	[null]
4	4	Cucumber	[null]	[null]

The left join starts selecting data from the left table. It compares values in the fruit_a column with the values in the fruit_b column in the basket_b table.

If these values are equal, the left join creates a new row that contains columns of both tables and adds this new row to the result set. (see the row #1 and #2 in the result set).

In case the values do not equal, the left join also creates a new row that contains columns from both tables and adds it to the result set. However, it fills the columns of the right table (`basket_b`) with null. (see the row #3 and #4 in the result set).

The following Venn diagram illustrates the left join:

LEFT OUTER JOIN
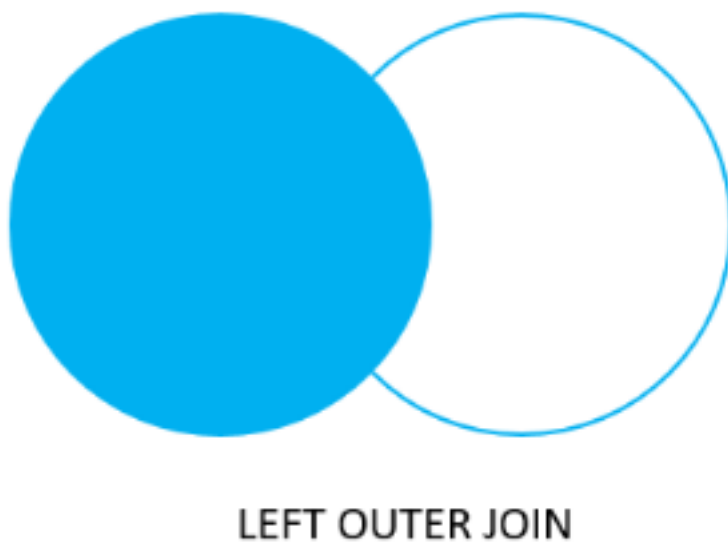
To select rows from the left table that do not have matching rows in the right table, you use the left join with a `WHERE` (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-where/) clause. For example:

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
LEFT JOIN basket_b
```

```
        ON fruit_a = fruit_b
    WHERE b IS NULL;
```

The output is:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	3	Banana	[null]	[null]
2	4	Cucumber	[null]	[null]

> Note that the `LEFT JOIN` is the same as the `LEFT OUTER JOIN` so you can use them interchangeably.

The following Venn diagram illustrates the left join that returns rows from the left table that do not have matching rows from the right table:
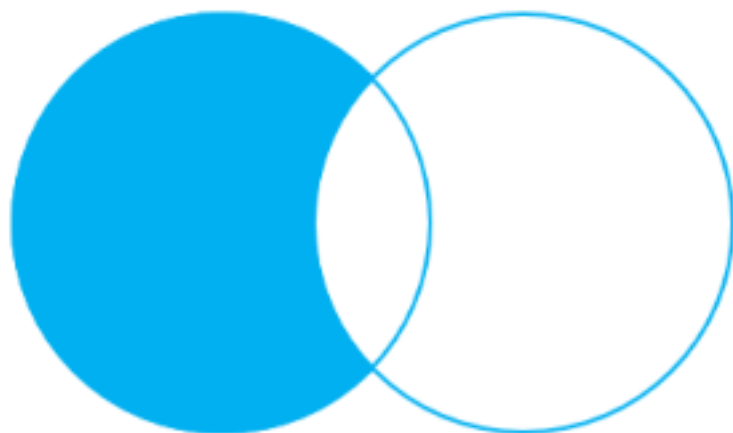
LEFT OUTER JOIN — only
rows from the left table

PostgreSQL right join

The right join (https://www.postgresqltutorial.com/postgresql-right-join/) is a reversed version of the left join. The right join starts selecting data from the right table. It compares each value in the fruit_b column of every row in the right table with each value in the fruit_a column of every row in the fruit_a table.

If these values are equal, the right join creates a new row that contains columns from both tables.

In case these values are not equal, the right join also creates a new row that contains columns from both tables. However, it fills the columns in the left table with NULL.

The following statement uses the right join to join the `basket_a` table with the `basket_b` table:

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
RIGHT JOIN basket_b ON fruit_a = fruit_b;
```

Here is the output:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	2	Orange	1	Orange
2	1	Apple	2	Apple
3	[null]	[null]	3	Watermelon
4	[null]	[null]	4	Pear

The following Venn diagram illustrates the right join:
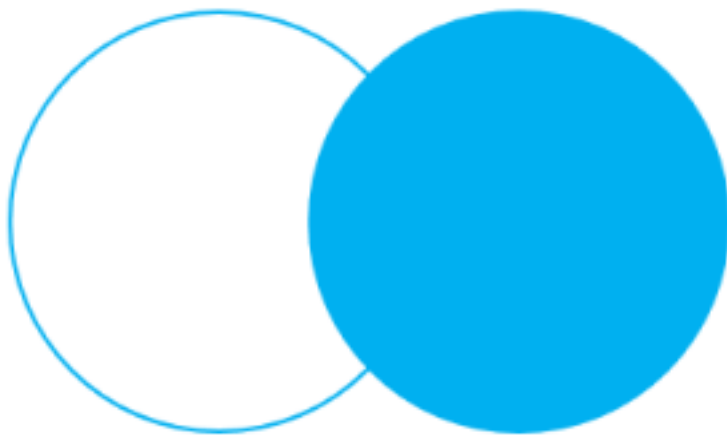
RIGHT OUTER JOIN

Similarly, you can get rows from the right table that do not have matching rows from the left table by adding a `WHERE` clause as follows:

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
RIGHT JOIN basket_b
    ON fruit_a = fruit_b
WHERE a IS NULL;
```

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	[null]	[null]	3	Watermelon
2	[null]	[null]	4	Pear

The `RIGHT JOIN` and `RIGHT OUTER JOIN` are the same therefore you can use them interchangeably.

The following Venn diagram illustrates the right join that returns rows from the right table that do not have matching rows in the left table:
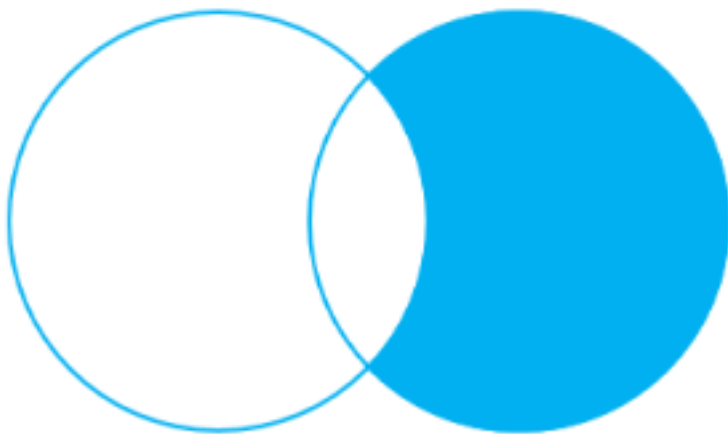
RIGHT OUTER JOIN – only
rows from the right table

PostgreSQL full outer join

The full outer join (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-full-outer-join/) or full join returns a result set that contains all rows from both left and right tables, with the matching rows from both sides if available. In case there is no match, the columns of the table will be filled with NULL.

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
FULL OUTER JOIN basket_b
    ON fruit_a = fruit_b;
```

Output:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange
3	3	Banana	[null]	[null]
4	4	Cucumber	[null]	[null]
5	[null]	[null]	3	Watermelon
6	[null]	[null]	4	Pear

The following Venn diagram illustrates the full outer join:

FULL OUTER JOIN

To return rows in a table that do not have matching rows in the other, you use the full join with a WHERE clause like this:

```
SELECT
    a,
    fruit_a,
    b,
    fruit_b
FROM
    basket_a
FULL JOIN basket_b
    ON fruit_a = fruit_b
WHERE a IS NULL OR b IS NULL;
```
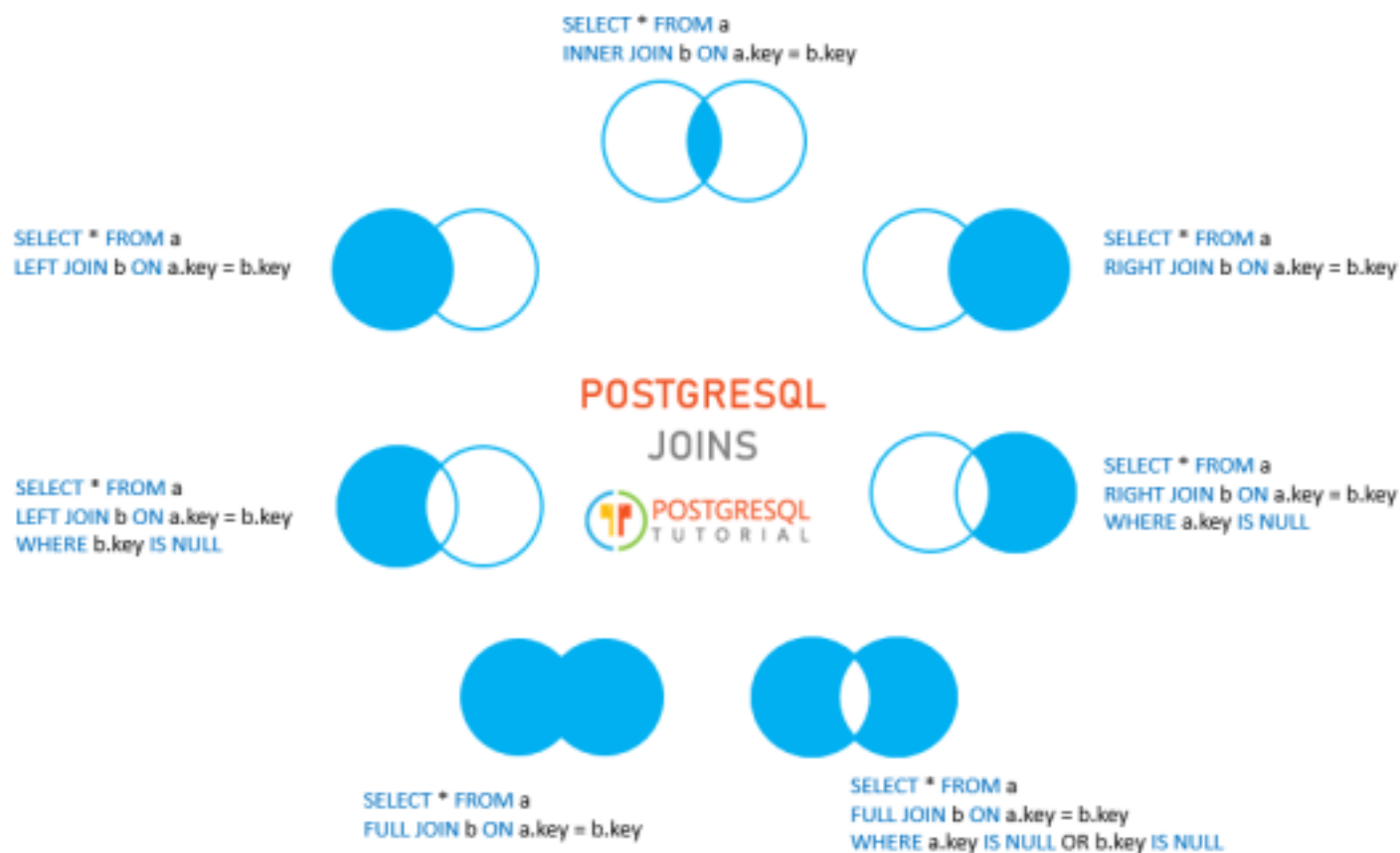
Here is the result:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	3	Banana	[null]	[null]
2	4	Cucumber	[null]	[null]
3	[null]	[null]	3	Watermelon
4	[null]	[null]	4	Pear

The following Venn diagram illustrates the full outer join that returns rows from a table that do not have the corresponding rows in the other table:

FULL OUTER JOIN – only
rows unique to both tables
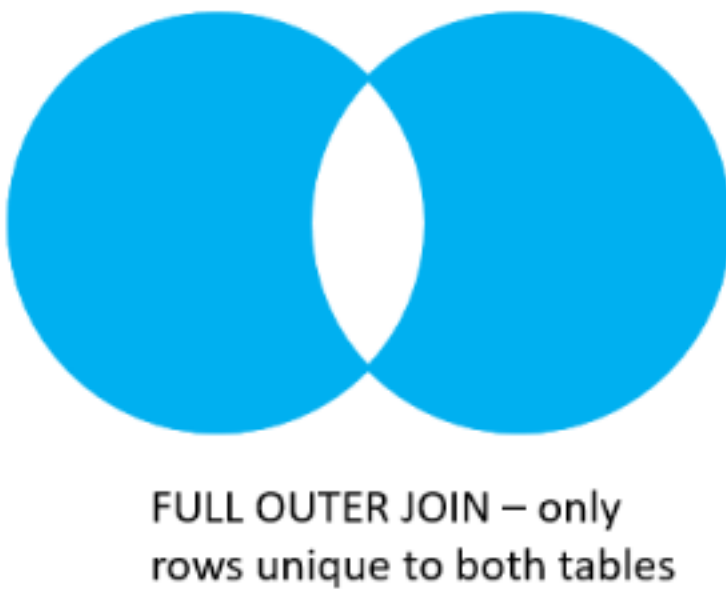
The following picture shows all the PostgreSQL joins that we discussed so far with the detailed syntax:

SELECT * FROM a
INNER JOIN b ON a.key = b.key

SELECT * FROM a
LEFT JOIN b ON a.key = b.key

SELECT * FROM a
RIGHT JOIN b ON a.key = b.key

POSTGRESQL
JOINS

POSTGRESQL
TUTORIAL

SELECT * FROM a
LEFT JOIN b ON a.key = b.key
WHERE b.key IS NULL

SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
WHERE a.key IS NULL

SELECT * FROM a
FULL JOIN b ON a.key = b.key

SELECT * FROM a
FULL JOIN b ON a.key = b.key
WHERE a.key IS NULL OR b.key IS NULL

In this tutorial, you have learned how to use various kinds of PostgreSQL joins to combine data from multiple related tables.