An Illustrated Guide to the SQL CROSS JOIN

Dorota Wdzięczna
Dorota is an IT engineer and works as a Data Science Writer for Vertabelo. She has experience as a Java programmer, webmaster, teacher, lecturer, IT specialist, and coordinator of IT systems. In her free time, she loves working in the garden, taking photos of nature, especially macro photos of insects, and visiting beautiful locations in Poland.

Tags: JOIN SQL CLAUSES TIP

What is an SQL CROSS JOIN statement? When should you use it? When shouldn't you use it? This post will tell you what you need to know about CROSS JOIN.

You already know that you can use the SQL `JOIN` statement to join one or more tables that share a matching record. And if you're read the LearnSQL's post LEARNING SQL JOINS USING REAL LIFE SITUATIONS, you know that there are many types of `JOIN`s. Which one you choose will depend on what you want it to do.

There's one `JOIN` that we don't use all that often but which serves a very specific purpose: the `CROSS JOIN`. In this article, I'll explain what a CROSS JOIN does and how it works. I'll also explain when you should use one, and when you shouldn't.

What is a JOIN?

First, let's review what a `JOIN` does in SQL: allows you to combine data (i.e. records) from multiple tables. `JOIN` operators let you join records in specific ways, such as only records that have a match in both tables.

As we start exploring `CROSS JOIN`, we will work begin with two tables, "`color`" and "`tshirt`".

The "`color`" table, which stores the names of available t-shirt colors, looks like this:

id	name
1	yellow
2	green
3	pink

The "`tshirt`" table, which stores the sizes of various t-shirts, looks like this:

id	size
1	S
2	M
3	L
4	XL

People wear t-shirts in all combinations of sizes and colors. So we will join data from these tables.

> Do you want to learn all the basics of SQL in one place? Go through our SQL From A to Z track!

What Is a CROSS JOIN?

`CROSS JOIN` returns a Cartesian product, or all records joined to all records in all tables. There is no `JOIN` condition (i.e. no ON clause). The resulting number of records equals the number of rows in the first table multiplied by the number of rows of the second table. `CROSS JOIN` is used very rarely. Because it produces all possible

combinations of records from all tables, it can be a dangerous operation for tables that contain a lot of records.

We have our two tables, shown below. Let's use a `CROSS JOIN` on them and see what happens:

tshirt

id	size
1	S
2	M
3	L
4	XL

color

id	name
1	yellow
2	green
3	pink

```
SELECT *  FROM  tshirt
CROSS JOIN  color;
```

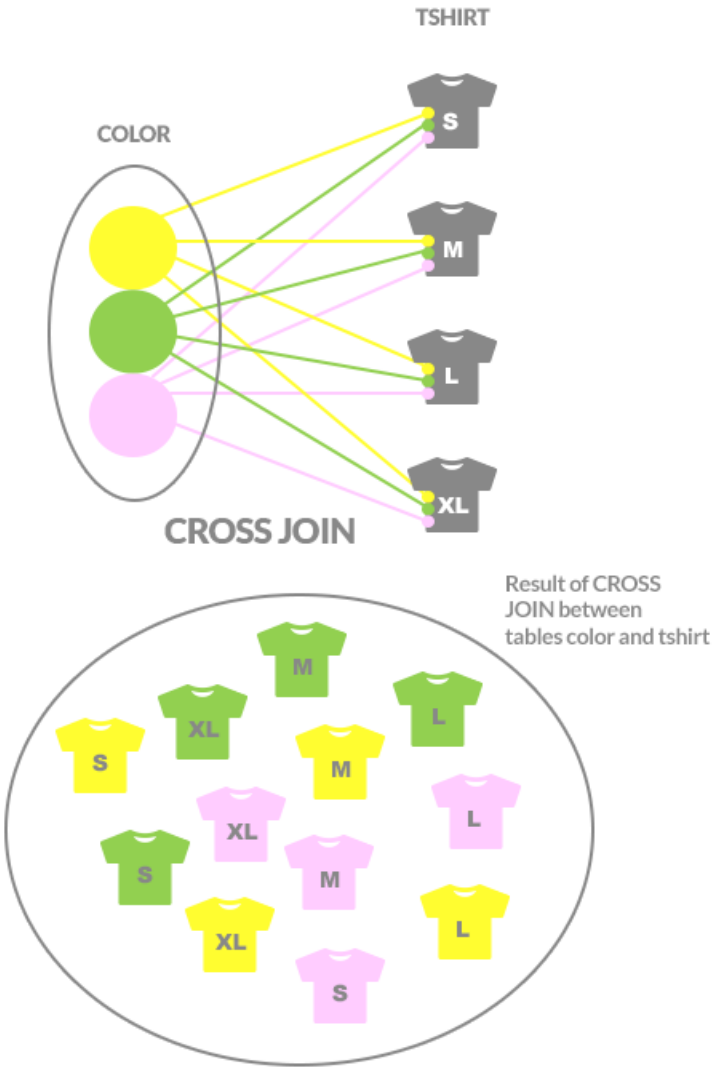
Notice that one table is listed after the `FROM` clause; the other table name follows `CROSS JOIN` It does not matter which table is listed after the from and which is after the `CROSS JOIN`. The results will be the same: all possible combination of records from all tables.

Here is the result of this operation:

id	size	id	name
1	S	1	yellow
2	M	1	yellow
3	L	1	yellow
4	XL	1	yellow
1	S	2	green

2	M	2	green
3	L	2	green
4	XL	2	green
1	S	3	pink
2	M	3	pink
3	L	3	pink
4	XL	3	pink

We've selected all records from the "`tshirt`" table and join each record with all the records from the "`color`" table. In other words, we have the combination of each T-shirt in each size with each color. The graphic below visually explains this:



Other Ways to Get Cartesian Products

There is another way to get a query to return a Cartesian product without using `CROSS JOIN`:

```
SELECT *  FROM  tshirt, color ;
```

In this case, **the tables you want to join must be listed after the `FROM` clause**. Note that by using the *, you are selecting all records from both tables. The result set will be the same as the `CROSS JOIN` example above.

> Want to know how to use JOINs? Check out our interactive SQL JOINs course.

Using CROSS JOIN in Multiplication

We can also use `CROSS JOIN` to do multiplication. Let's use two tables that store some basic numbers to see how this works. First, have a look at the tables:

"t1"

n
11
12
13

"t2"

n
11
12
13

Both tables store the same numbers: 11, 12, 13. Our goal is multiply these numbers by themselves, creating a multiplication table of sorts. We can use **CROSS JOIN** to do this. Look at the query below:

```
SELECT t1.n, t2.n, t1.n*t2.n AS result  FROM  t1
CROSS JOIN  t2 ;
```

The `SELECT` lists three columns: numbers from the first table `(t1.n)`, numbers from the second table `(t2.n)`, and the multiplication that is stored in the result column `(t1.n*t2.n AS result)`. The table below shows the outcome:

t1.n	t2.n	result
11	11	121
11	12	132
11	13	143
12	11	132
12	12	144
12	13	156
13	11	143
13	12	156
13	13	169

Using CROSS JOIN with Many Tables

We can use more than two tables in a `CROSS JOIN`. Imagine that a company produces t-shirts in three sizes (S, M, L) and in two fabrics (cotton and linen). Each

t-shirt is available in two colors: pink or blue. Have a look at the following tables:

"color"

id	name
1	blue
3	pink

"tshirt"

id	size
1	S
2	M
3	L

"fabric"

id	name
1	cotton
2	linen

Now suppose you want to see all possible t-shirts: every combination of size, color, and fabric. The query below retrieves this information using a `CROSS JOIN` on three tables:

```
SELECT tshirt.size as size, color.name AS color, fabric.name as fabric
FROM  tshirt
CROSS JOIN  fabric
CROSS JOIN  color ;
```

Here is the result:

size	color	fabric
S	blue	cotton
M	blue	cotton
L	blue	cotton
S	pink	cotton
M	pink	cotton
L	pink	cotton
S	blue	linen
M	blue	linen
L	blue	linen
S	pink	linen
M	pink	linen
L	pink	linen

There are twelve possible t-shirts. The two colors multiplied by three sizes and again by two fabrics result in a 12.

Learn More

There's more to know about using `CROSS JOIN` than what we've covered in this illustrated introduction. If you want to learn more about `JOIN`s, check out LearnSQL's SQL BASICS COURSE.