CROSS JOINs

by Sophia

WHAT'S COVERED

In this lesson, you will explore using CROSS JOINs to query data from tables. Specifically, this lesson will cover:

1. CROSS JOINs
2. Examples With Simple Tables

1. CROSS JOINs

CROSS JOINs are SQL operations that combine every row of one table with every row of another. A CROSS JOIN creates a new table with the same number of rows as the product of the original table sizes, which differs from other types of JOINs that match conditions. It is also called a cross product or CARTESIAN JOIN since all row combinations are included in the result. One possible use of a CROSS JOIN is to explore all possible combinations between two data sets.

This JOIN type differs from other JOIN types since it does not require specific conditions for joining. Although CROSS JOINs are powerful tools for generating all possible combinations of rows between two tables, they should be used sparingly. CROSS JOINs can result in a huge number of rows being returned without the proper filtering mechanisms. Data overload and performance issues may result from this.

Because CROSS JOIN can result in large data sets, it's helpful to apply filters or combine other JOIN types with it. By doing this, you can narrow your search results to only show the information you seek.

The structure of the CROSS JOIN looks like this:

```
SELECT <columnlist>
FROM <table1>
CROSS JOIN <table2>;
```

Notice that there are no details of how the tables are joined. This statement would result in the same result set as if we did the following:

```
SELECT <columnlist>
FROM <table1>,<table2>;
```

Or, we could even use an INNER JOIN with a condition that always evaluates to true to force a CROSS JOIN:

```
SELECT <columnlist>
FROM <table1>
INNER JOIN <table2> ON true;
```

TERM TO KNOW

CROSS JOIN

A clause in a query that creates a new table with the same number of rows as the product of the original table sizes.

2. Examples With Simple Tables

Let's try this by creating a couple of tables for size and color and joining them together.

```
CREATE TABLE color ( color_id INT PRIMARY KEY, color_name VARCHAR (50) NOT NULL );
CREATE TABLE size ( size_id INT PRIMARY KEY, size_name VARCHAR (30) NOT NULL );
INSERT INTO color (color_id, color_name)
VALUES (1, 'Blue'), (2, 'Red'), (3, 'Yellow');
INSERT INTO size (size_id, size_name)
VALUES (1, 'Small'), (2, 'Medium'), (3, 'Large');
```

Now that the tables have been created, we should expect to see nine rows returned from the CROSS JOIN, as there are three rows in the color table and three in the size table:

```
SELECT color_id, color_name, size_id, size_name
FROM color
CROSS JOIN size;
```

Query Results

Row count: 9

color_id	color_name	size_id	size_name
1	Blue	1	Small
2	Red	1	Small
3	Yellow	1	Small
1	Blue	2	Medium
2	Red	2	Medium
3	Yellow	2	Medium
1	Blue	3	Large
2	Red	3	Large
3	Yellow	3	Large

|alt=Query Results Cross Join Color ID, Color Name, Size ID, Size Name]]

To understand the inner workings of CROSS JOIN, it's helpful to look at what is happening behind the scenes. You could run the following statement to do the same thing that CROSS JOIN does much more compactly.

```
SELECT color_id, color_name, size_id, size_name
FROM color, size;
SELECT color_id, color_name, size_id, size_name
FROM color
INNER JOIN size ON true;
```

Query Results

Row count: 9

color_id	color_name	size_id	size_name
1	Blue	1	Small
2	Red	1	Small
3	Yellow	1	Small
1	Blue	2	Medium
2	Red	2	Medium
3	Yellow	2	Medium
1	Blue	3	Large
2	Red	3	Large
3	Yellow	3	Large

As you can see, the CROSS JOIN has taken each color and matched it to each size to display all possible combinations. This type of combination can become problematic as the size of the data gets larger. In our example, we only have three rows in the color table and three rows in the size table. But imagine if we had 1,000 rows in one table and 1,000 rows in another table. The result would have 1,000,000 rows in the result. This is why CROSS JOIN is useful only in rare, specific scenarios.

WATCH

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

In this lesson, you learned that a **CROSS JOIN**, also called a CARTESIAN JOIN, combines every row in a table with every row in another table. There are no specific criteria, unlike with other types of JOINs. CROSS JOINs are useful when you need to generate all possible combinations of rows from two simple tables. Use them with caution, as they can quickly result in a large result set. You can reduce the result set size by applying filtering conditions or combining with other JOIN types.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

TERMS TO KNOW

CROSS JOIN

A clause in a query that creates a new table with the same number of rows as the product of the original table sizes.