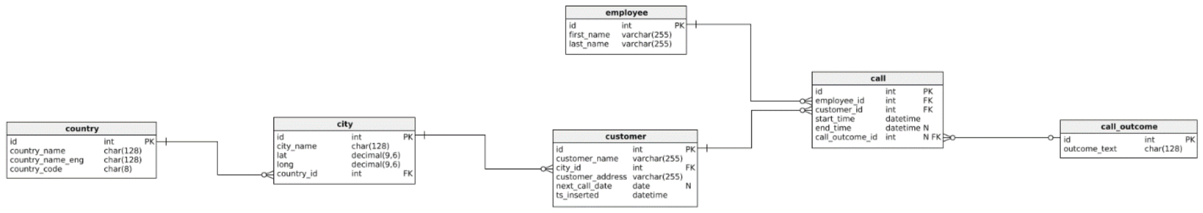
Learn SQL: Join multiple tables

January 27, 2020 by Emil Drkusic

If you want to get something meaningful out of data, you'll almost always need to join multiple tables. In this article, we'll show how to do that using different types of joins. To achieve that, we'll combine INNER JOINs and LEFT JOINs. So, let's start.

The Model

In the picture below you can see out existing model. It consists of 6 tables and we've already, more or less, described it in the previous articles.



Still, even without describing, if the database is modeled and presented in a good manner (choosing names wisely, using naming convention, following the same rules throughout the whole model, lines/relations in schema do not overlap more than needed), you should be able to conclude where you can find the data you need. This is crucial because before you join multiple tables, you need to

In this series, we've covered:

- Basics related to SQL SELECT statement, and
- Compared INNER JOIN and LEFT JOIN

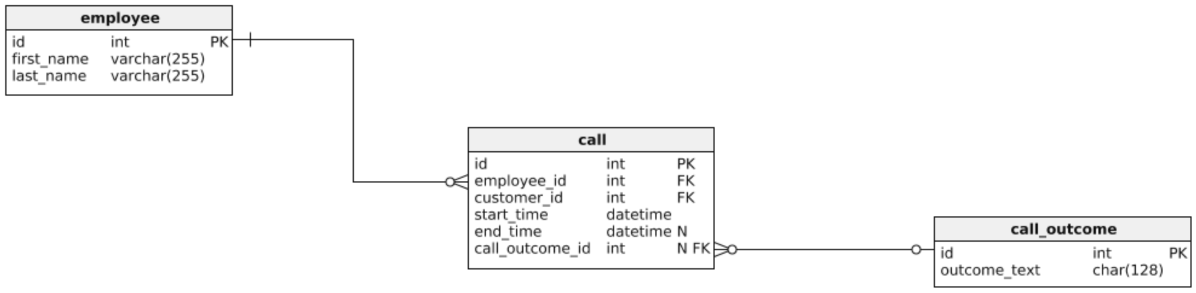
We'll use the knowledge from both these articles and combine these to write more complex SELECT statements that will join multiple tables.

Join multiple tables using INNER JOIN

The first example we'll analyze is how to retrieve data from multiple tables using only INNER JOINs. For each example, we'll go with the definition of the problem we must solve and the query that does the job. So, let's start with the first problem.

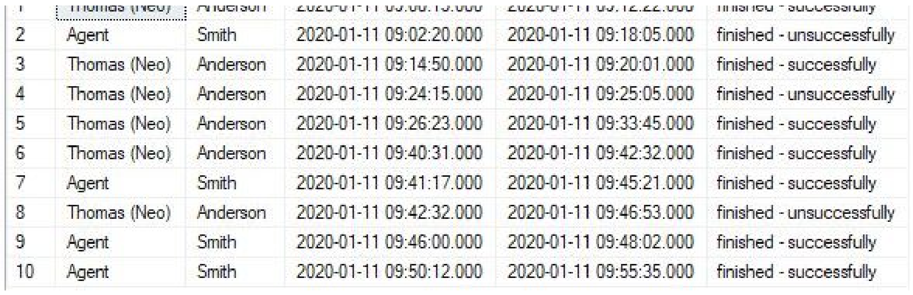
#1 We need to list all calls with their start time and end time. For each call, we want to display what was the outcome as well the first and the last name of the employee who made that call. We'll sort our calls by start time ascending.

Before we write the query, we'll identify the tables we need to use. To do that, we need to determine which tables contain the data we need and include them. Also, we should include all tables along the way between these tables – tables that don't contain data needed but serve as a relation between tables that do (that is not the case here).



The query that does the job is given below:

```
SELECT employee.first_name, employee.last_name, call.start_time, call.end_time, ca
```



1	Thomas (Neo)	Anderson	2020-01-11 09:00:15.000	2020-01-11 09:12:22.000	finished - successfully
2	Agent	Smith	2020-01-11 09:02:20.000	2020-01-11 09:18:05.000	finished - unsuccessfully
3	Thomas (Neo)	Anderson	2020-01-11 09:14:50.000	2020-01-11 09:20:01.000	finished - successfully
4	Thomas (Neo)	Anderson	2020-01-11 09:24:15.000	2020-01-11 09:25:05.000	finished - unsuccessfully
5	Thomas (Neo)	Anderson	2020-01-11 09:26:23.000	2020-01-11 09:33:45.000	finished - successfully
6	Thomas (Neo)	Anderson	2020-01-11 09:40:31.000	2020-01-11 09:42:32.000	finished - successfully
7	Agent	Smith	2020-01-11 09:41:17.000	2020-01-11 09:45:21.000	finished - successfully
8	Thomas (Neo)	Anderson	2020-01-11 09:42:32.000	2020-01-11 09:46:53.000	finished - unsuccessfully
9	Agent	Smith	2020-01-11 09:46:00.000	2020-01-11 09:48:02.000	finished - successfully
10	Agent	Smith	2020-01-11 09:50:12.000	2020-01-11 09:55:35.000	finished - successfully

There are a few things I would like to point out here:

- The tables we've joined are here because the data we need is located in these 3 tables
- Each time I mention any attribute from any table, I'm using format table_name.attribute_name (e.g. **employee.first_name**). While that's not needed, it's a good practice, because sometimes 2 or more tables in the same query could use the same attribute names and that would lead to an error
- We've used INNER JOIN 2 times in order to join 3 tables. This will result in returning only rows having pairs in another table
- When you're using only INNER JOINs to join multiple tables, the order of these tables in joins is not important. The only important thing is that you use appropriate join conditions after the "ON" (join using foreign keys)

Since all calls had related employee and call outcome, we would get the same result if we've used LEFT JOIN instead of the INNER JOIN.

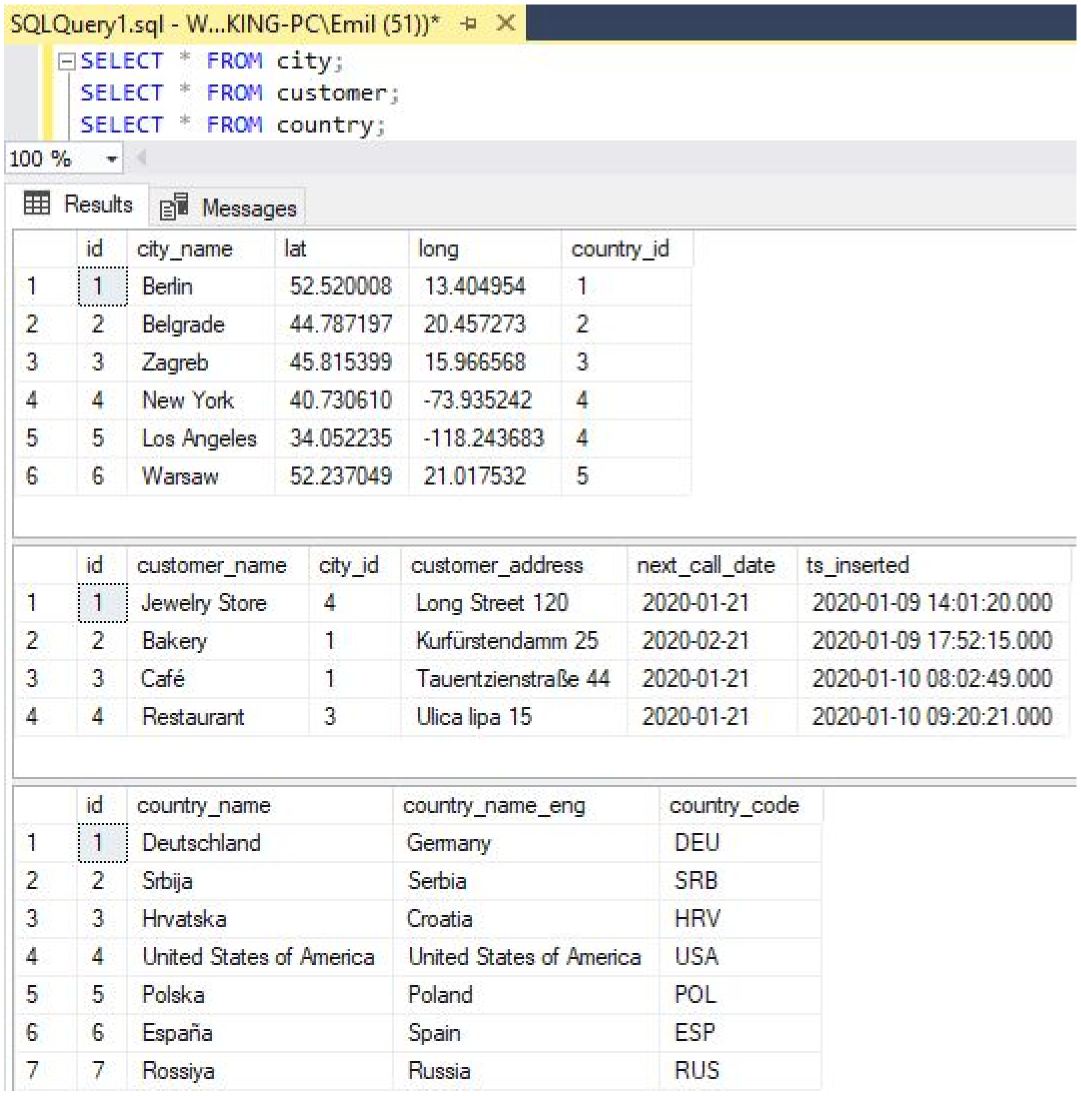
Join multiple tables using LEFT JOIN

Writing queries that use LEFT JOINs doesn't differ a lot when compared to writing queries using INNER JOINs. The result would, of course, be different (at least in cases when some records don't have a pair in other tables).

This is the problem we want to solve.

#2 List all counties and customers related to these countries. For each country display its name in English, the name of the city customer is located in as well as the name of that customer. Return even countries without related cities and customers.

First, let's quickly check what is the contents of these 3 tables.


SQLQuery1.sql - W...KING-PC\Emil (51))*

```sql
SELECT * FROM city;
SELECT * FROM customer;
SELECT * FROM country;
```

	id	city_name	lat	long	country_id
1	1	Berlin	52.520008	13.404954	1
2	2	Belgrade	44.787197	20.457273	2
3	3	Zagreb	45.815399	15.966568	3
4	4	New York	40.730610	-73.935242	4
5	5	Los Angeles	34.052235	-118.243683	4
6	6	Warsaw	52.237049	21.017532	5

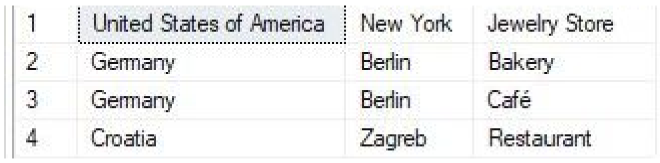
	id	customer_name	city_id	customer_address	next_call_date	ts_inserted
1	1	Jewelry Store	4	Long Street 120	2020-01-21	2020-01-09 14:01:20.000
2	2	Bakery	1	Kurfürstendamm 25	2020-02-21	2020-01-09 17:52:15.000
3	3	Café	1	Tauentzienstraße 44	2020-01-21	2020-01-10 08:02:49.000
4	4	Restaurant	3	Ulica lipa 15	2020-01-21	2020-01-10 09:20:21.000

	id	country_name	country_name_eng	country_code
1	1	Deutschland	Germany	DEU
2	2	Srbija	Serbia	SRB
3	3	Hrvatska	Croatia	HRV
4	4	United States of America	United States of America	USA
5	5	Polska	Poland	POL
6	6	España	Spain	ESP
7	7	Rossiya	Russia	RUS

We can notice two important things:

- While each **city** has a related **country**, not all countries have related cities (Spain & Russia don't have them)
- Same stands for the customers. Each **customer** has the **city_id** value defined, but only 3 cities are being used (Berlin, Zagreb & New York)

Let's first write down the query using INNER JOIN:



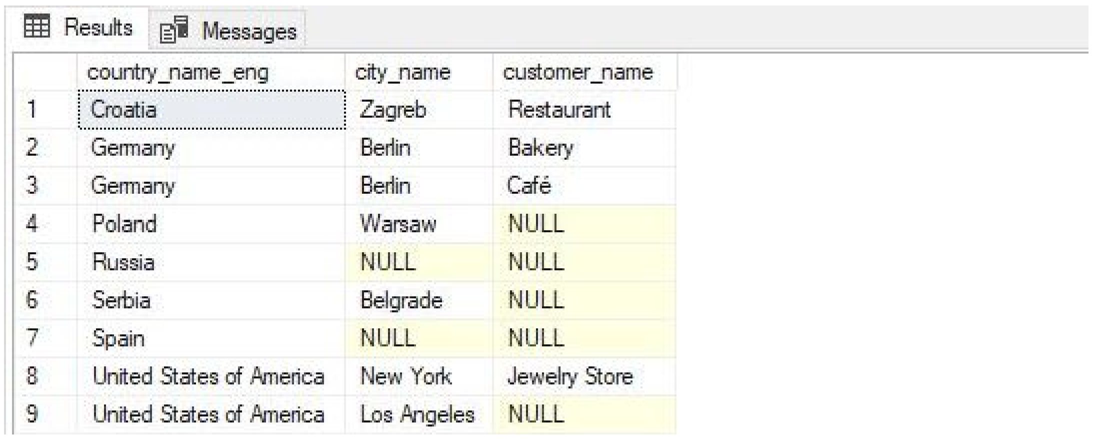
1	United States of America	New York	Jewelry Store
2	Germany	Berlin	Bakery
3	Germany	Berlin	Café
4	Croatia	Zagreb	Restaurant

We have 7 counties and 6 cities in our database, but our query returns only 4 rows. That is the result of the fact we have only 4 customers in our database. Each of these 4 is related to its city and the city is related to the country. So, INNER JOIN eliminated all these countries and cities without customers. But how to include these in the result too?

To do that, we'll use LEFT JOIN. We'll simply replace all "INNER" with "LEFT" so our query is as follows:

```
SELECT country.country_name_eng, city.city_name, customer.customer_name
FROM country
LEFT JOIN city ON city.country_id = country.id
LEFT JOIN customer ON customer.city_id = city.id;
```

The result is shown in the picture below:



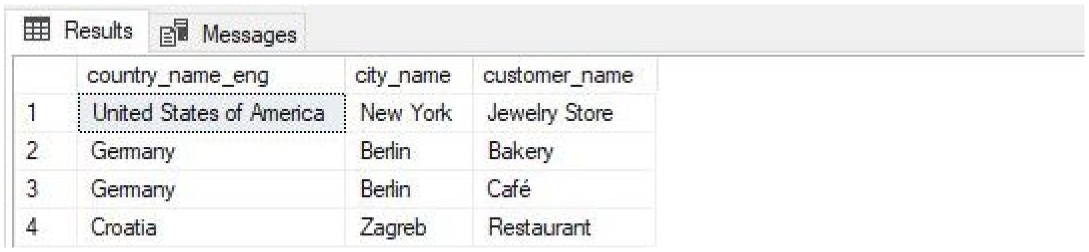
	country_name_eng	city_name	customer_name
1	Croatia	Zagreb	Restaurant
2	Germany	Berlin	Bakery
3	Germany	Berlin	Café
4	Poland	Warsaw	NULL
5	Russia	NULL	NULL
6	Serbia	Belgrade	NULL
7	Spain	NULL	NULL
8	United States of America	New York	Jewelry Store
9	United States of America	Los Angeles	NULL

You can easily notice that now we have all the countries, even those without any related city (Russia & Spain), as well all cities, even those without customers (Warsaw, Belgrade & Los Angeles). The remaining 4 rows are the same as in the query using INNER JOIN.

LEFT JOIN – Tables order matters

While the order of JOINs in INNER JOIN isn't important, the same doesn't stand for the LEFT JOIN.

at the output first:



	country_name_eng	city_name	customer_name
1	United States of America	New York	Jewelry Store
2	Germany	Berlin	Bakery
3	Germany	Berlin	Café
4	Croatia	Zagreb	Restaurant

So, what happened here? Why do we have 4 rows (same 4 we had when we've used INNER JOIN)?

The answer is simple and it's related to how LEFT JOIN works. It takes the first table (**customer**) and joins all its rows (4 of them) to the next table (**city**). The result of this is 4 rows because the customer could belong to only 1 city. Then we join these 4 rows to the next table (**country**), and again we have 4 rows because the city could belong to only 1 country.

The reason why we wouldn't join these 3 tables in this way is given by the text of the example #2. The query is written in such manner it returns 4 rows would be the answer to the following: Return names of all customers as well as cities and countries they are located in. Return even customers without related cities and countries.

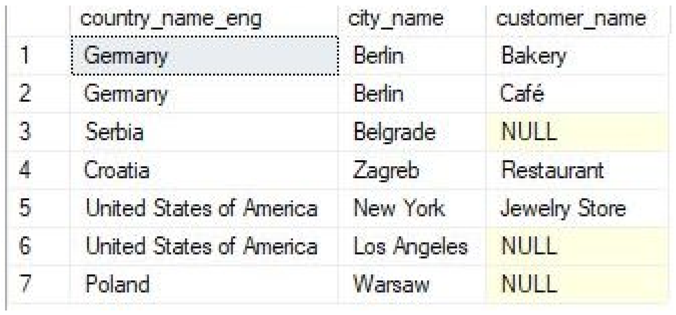
> **Note:** When you're using LEFT JOIN, the order of tables in that statement is important and the query will return a different result if you change this order. The order actually depends on what you want to return as a result.

Join multiple tables using both – INNER JOIN & LEFT JOIN

This is also possible. Let's again go with an example.

#3 Return the list of all countries and cities that have pair (exclude countries which are not referenced by any city). For such pairs return all customers. Return even pairs not having a single customer.

The query that does the job is:



	country_name_eng	city_name	customer_name
1	Germany	Berlin	Bakery
2	Germany	Berlin	Café
3	Serbia	Belgrade	NULL
4	Croatia	Zagreb	Restaurant
5	United States of America	New York	Jewelry Store
6	United States of America	Los Angeles	NULL
7	Poland	Warsaw	NULL

You can easily notice that we don't have countries without any related city (these were Spain & Russia). The INNER JOIN eliminated these rows. Still, we do have cites without any customers (Belgrade, Los Angeles & Warsaw). This is the result of the fact we used LEFT JOIN between tables **city** and **customer**.

Conclusion

When you need to join multiple tables, you have INNER & LEFT JOIN on your disposal (RIGHT JOIN is rarely used and can be easily replaced by LEFT JOIN). Which join you'll use depends directly on the task you need to solve and you'll get the feeling along the way. In upcoming articles, we'll discuss how to think and organize yourself when you need to write more complex queries.

Table of contents

See more

Emil Drkusic

Emil is a database professional with 10+ years of experience in everything related to databases. During the years, he worked in the IT and finance industry and now works as a freelancer.

His past and present engagements vary from database design and coding to teaching, consulting, and writing about databases. Also not to forget, BI, creating algorithms, chess, philately, 2 dogs, 2 cats, 1 wife, 1 baby…

You can find him on LinkedIn

View all posts by Emil Drkusic

203,284 Views

ALSO ON **SQL SHACK**

6 months ago • 1 comment
Split SQL database backups into …

5 months ago • 1 comment
Don't fear SQL Server performance

5 months a
Uploa data ir Blob .

0 Comments **SQL Shack** **Login**

Recommend 7 Tweet Share **Sort by Best**

Start the discussion