Joins

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores the needs to join tables together in various SQL statements in two parts:

1. Inner Joins
2. Outer Joins

1. Inner Joins

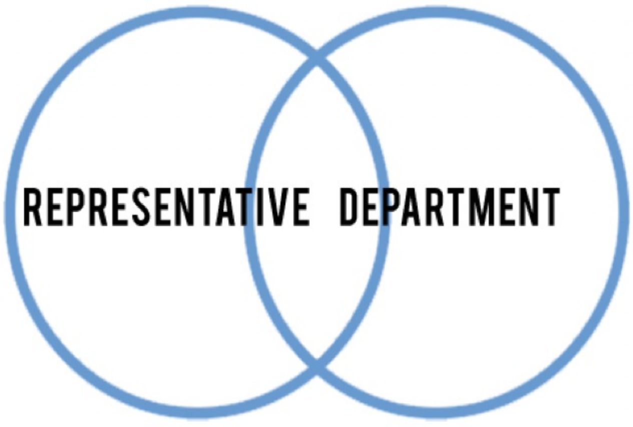
Up to this point, all the SQL statements we've been querying have been using a single table. However, databases really start to shine when we can join the data in tables together. Joins are done when you combine data from two or more tables, linking them through common attributes. There are different types of joins, including natural joins, equijoin, outer join, cross join and self joins, that we will work through in upcoming tutorials.

Let's first create two tables for representatives and departments. The common attribute between the two tables is the representative_id from the representative table and the manager_id in the manager table. Notice that in this case, the two column names are different from one another but linked through the primary key.

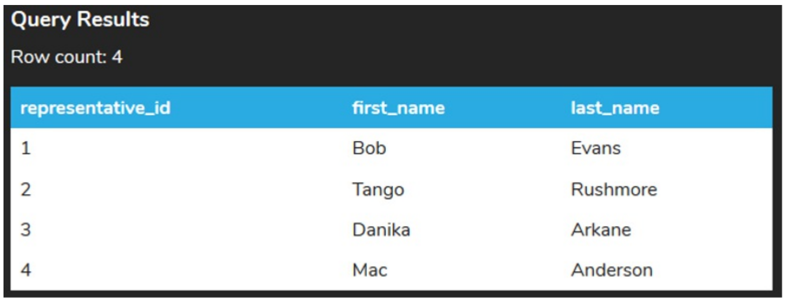
```
CREATE TABLE representative ( representative_id INT PRIMARY KEY, first_name VARCHAR (30) NOT NULL, last_name VARCHAR (30) NOT NULL );

CREATE TABLE department ( department_id INT PRIMARY KEY, department_name VARCHAR (100) NOT NULL, manager_id INT, constraint fk_manager FOREIGN KEY (manager_id) REFERE
);

INSERT INTO representative (representative_id, first_name, last_name)
VALUES (1, 'Bob','Evans'), (2, 'Tango','Rushmore'), (3, 'Danika','Arkane'), (4, 'Mac','Anderson');

INSERT INTO department (department_id, department_name,manager_id)
VALUES (1, 'Sales', 1), (2, 'Marketing', 3), (3, 'IT', 4), (4, 'Finance', null), (5, 'Support', null);
```

In looking at the data, we can see that Tango Rushmore is not a manager of a department, and the Finance and Support departments do not yet have a manager.

We can represent this data through the use of a Venn diagram, where the left circle represents the data from the representative table and the right circle represents the data from the department table:

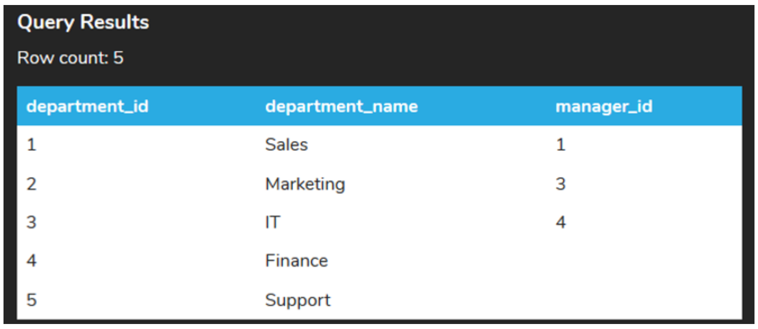


Individually, the data looks like the following:


Query Results

Row count: 4

representative_id	first_name	last_name
1	Bob	Evans
2	Tango	Rushmore
3	Danika	Arkane
4	Mac	Anderson


Query Results

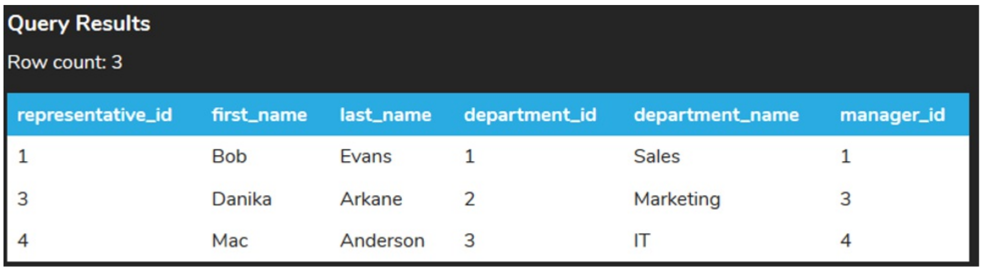
Row count: 5

department_id	department_name	manager_id
1	Sales	1
2	Marketing	3
3	IT	4
4	Finance	
5	Support	

If we had the same column name in the two tables, we could complete a natural join, which you will learn more about soon. Instead, we can complete an inner join by identifying the column name. The basic inner join between the two tables should look like the following:

```
SELECT *
FROM representative
JOIN department
ON representative.representative_id = department.manager_id;
```

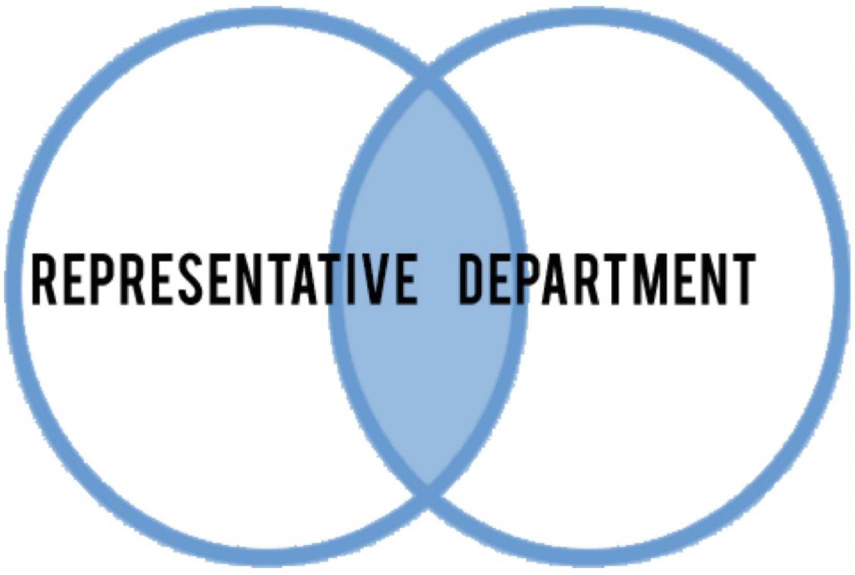
We will get into the specifics of the syntax in later tutorials. With this inner join, we find that the data in the representative_id column in the representative table matches the department's manager_id. There should be three that match:


Query Results

Row count: 3

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4

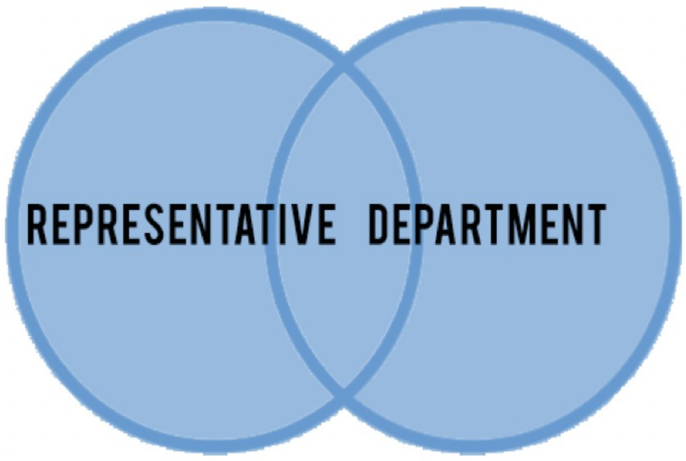
The Venn diagram below illustrates the inner join:

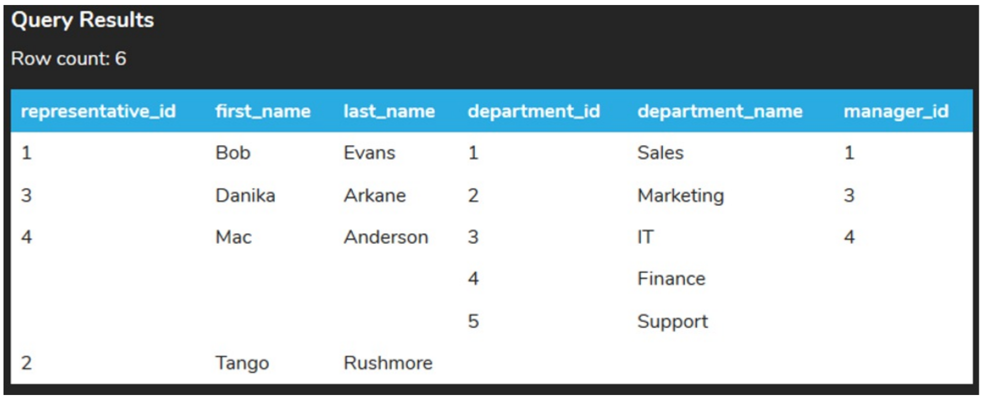


This is the most common type of join, as we typically want to be able to identify where data between two or more tables matches.

2. Outer Joins

The OUTER join allows us to get data that may not fully match between two tables:

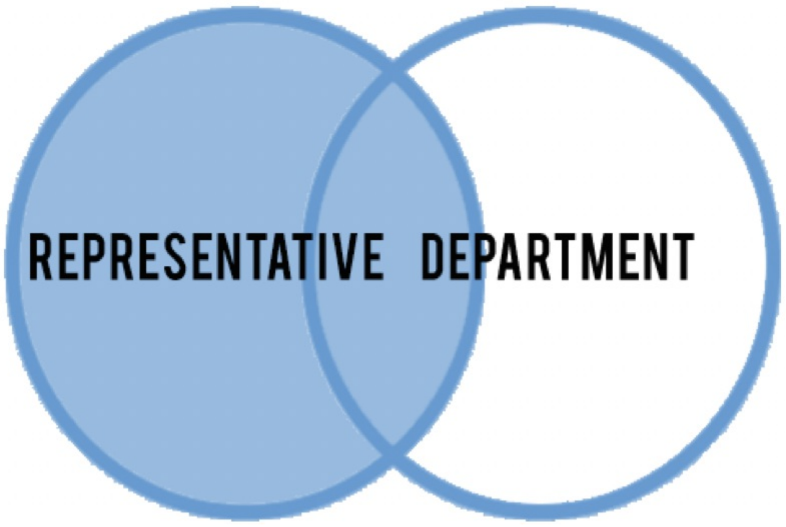


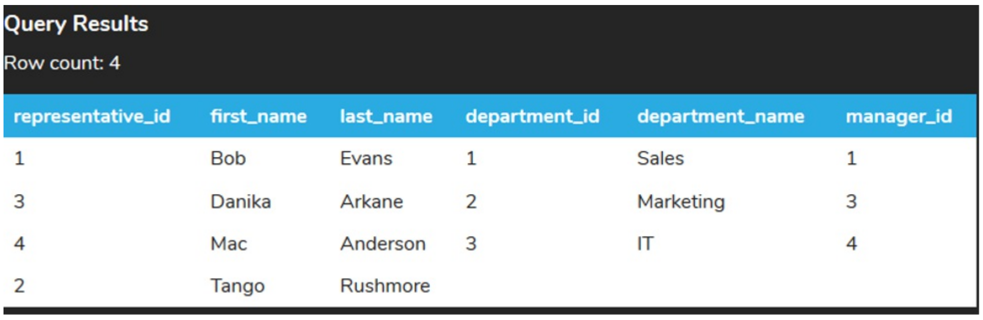

Query Results

Row count: 6

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4
			4	Finance	
			5	Support	
2	Tango	Rushmore			

We may want to use a left join to find data that exists in both tables, but also data in the left table that does not match. This Venn diagram illustrates a left join.

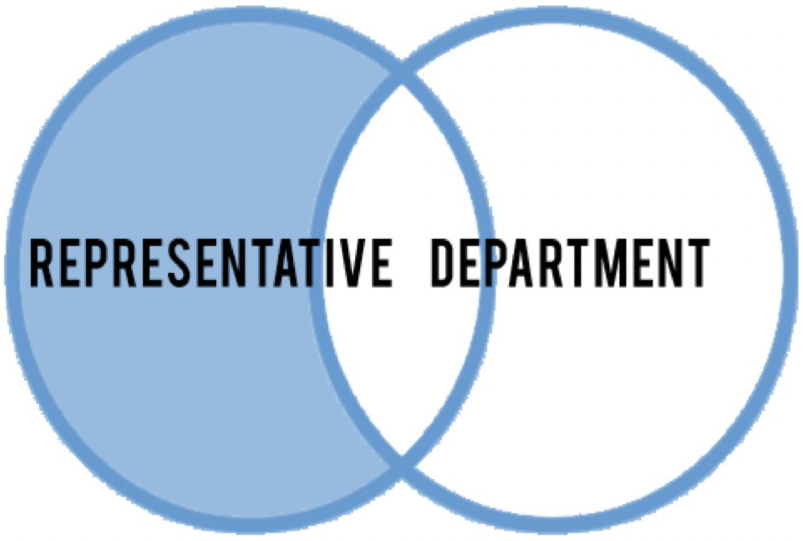


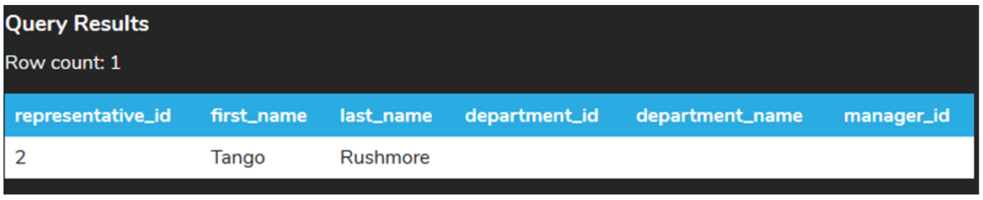

Query Results

Row count: 4

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4
2	Tango	Rushmore			

We may also want to find data that exists in the left table, but does not match with data in the right table. This would be considered a left outer join:

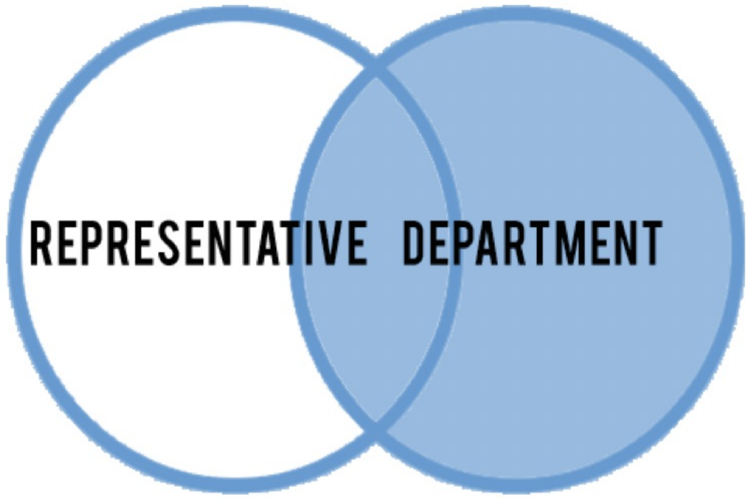


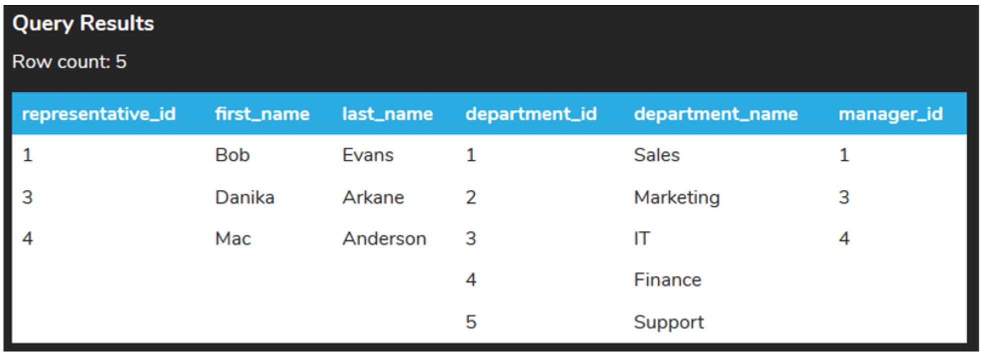

Query Results

Row count: 1

representative_id	first_name	last_name	department_id	department_name	manager_id
2	Tango	Rushmore			

Using a right join, we can get the data that exists in both tables, along with the data that does not match from the right table:

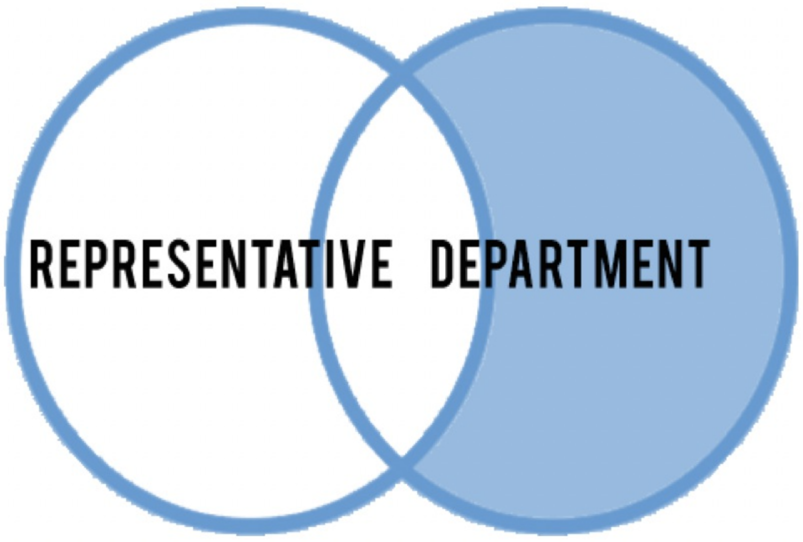


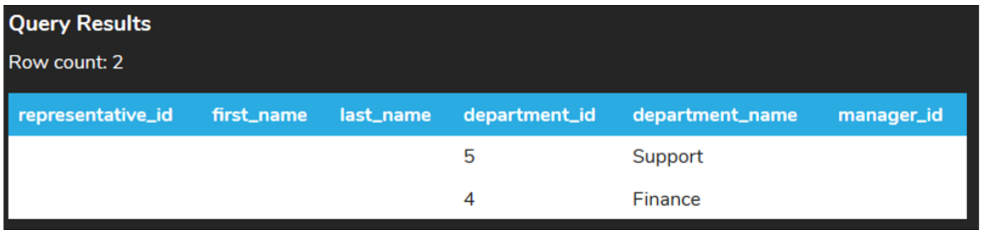

Query Results

Row count: 5

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4
			4	Finance	
			5	Support	

We can also find data that exists in the right table, but does not match with data in the left table, using a right outer join:




Query Results

Row count: 2

representative_id	first_name	last_name	department_id	department_name	manager_id
			5	Support	
			4	Finance	

Video Transcription

[MUSIC PLAYING] So far up to this point, we've mainly focused on single queries that query a single table. However, where sequel and databases in general really come as beneficial is when you start querying data together through the use of joins. So in this case here, we're going to have two different tables-- the representative table and the department table.

In this case, the manager ID that's within the department table is a foreign key to the representative ID in the representative table. So any data that's inserted into department references to the representative table to ensure that that value does exist on the department table.

So as part of this, we can go ahead and run a specific type of join called an inner join. There's multiple different types of inner joins, including natural joins, adjoining on, joining using, and so forth. It is important to differentiate them and we'll get into that as we progress through the unit. So in this case here, what happens with the selection in this case is that it links the two tables together on their representative ID where it matches to whatever the manager ID is. It only returns a record setting match between the two and any records doesn't match doesn't get returned.

What we also have are outer joins, and outer joins are as follows. And this one in this case here, is going to be a full outer join, meaning that it's going to list all the different rows and match as you see here, as a starting point, but we'll also see the rows. That's in the department table that don't have any matching data on the representative table as well as no data that's in the representative table, but not matching to what is in the department table.

Throughout the unit, we'll explore lots of different types of joins, including the full outer joins, the left join, the right join, cross joins, and the other inner joins.

[MUSIC PLAYING]

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

We can use inner and outer joins to link table data together using common attributes.