Joining Related Tables

Introduction to SQL Joins

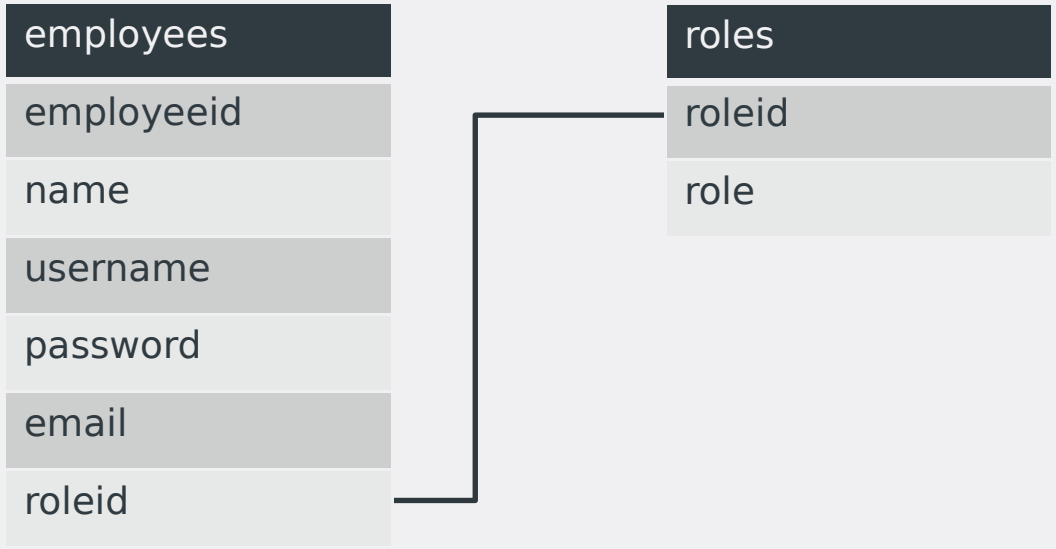
Up to this point, we've focused primarily on extracting data from a single table. Depending on how advanced your database becomes, at times you might want to extract data from multiple tables at once. If that's the case, you will need to use joins. Although there are several types of joins, two types will be covered here:

- INNER JOIN
- OUTER JOIN
- LEFT JOIN (not covered)
- RIGHT JOIN (not covered)
- FULL JOIN (not covered)
- SELF JOIN (not covered)
- UNION (not covered)

Joining Related Tables

Introduction to INNER JOIN

Of the different types of joins, the inner join is by far the most popular. An Inner join enables you to see all the records of two tables that have relationships established to one another within a single query. Remember that the employees and the roles tables have an established one-to-many relationship. The two tables in the database resemble the following:



Joining Related Tables

Introduction to INNER JOIN

Assume that you wanted to extract the information from the employees table for employeeid 2. Your SELECT statement would resemble the following:

```
SELECT * FROM employees WHERE employeeid = 2
```

This statement would produce the following result:

employeeid	name	username	password	email	roleid
2	Wilbur	wfounder	password	wilbur@vectacorp.com	2

Although the results are shown just fine, the issue resides within the roleid field. What exactly does 2 mean? The numeric value of 2 doesn't mean much to the end user. Instead we would want to extract the actual role of *Support* or *Developer* directly from the roles table. This is where the INNER JOIN command comes in.

Joining Related Tables

Introduction to INNER JOIN

By modifying the SQL statement slightly, we can easily extract data from both the employees and roles tables at the same time:

```
SELECT employees.*, roles.* FROM employees
INNER JOIN roles ON employees.roleid = roles.roleid
```

employeeid	name	username	password	email	roleid	role
2	Wilbur	wfounder	password	wilbur@vectacorp.com	2	Support

Because of the INNER JOIN, we are able to effectively query both the employees and roles tables and return the data into a single, virtual result set.

Joining Related Tables

Introduction to INNER JOIN

Notice that the preceding table now becomes more efficient and manageable. Also notice that, rather than referencing the names of the tables, you used the tableName.fieldname notation. This is crucial when using joins; otherwise, you would end up with two employeeids without a direct reference to its corresponding table.

Also note the use of the ON operator in the preceding INNER JOIN statement. The ON operator instructs the statement to join two tables on a specific primary and foreign key pairing.

Joining Related Tables

Introduction to OUTER JOIN

Outer joins enable rows to be returned from a join in which one of the tables does not contain matching rows for the other table. As an example, suppose the roleid field in the employees table wasn’t a required field. For instance, Tina is clearly a support technician and Damon and Wally are developers. The other employees however, may not have roles listed within the employees table. You could end up with data that resembled the following:

employeeid	name	username	password	email	roleid
1	Wally	wwebmaster	password	wally@vectacorp.com	2
2	Wilbur	wfounder	password	wilbur@vectacorp.com	
3	Tina	ttechie	abc123	tina@vectacorp.com	1
4	Agnes	aaccountant	12345	agnes@vectacorp.com	
5	Damon	ddeveloper	ispeakbinary	damon@vectacorp.com	2

Joining Related Tables

Introduction to OUTER JOIN

The OUTER JOIN would be a perfect choice when attempting to join two tables where the foreign key field may not be required. The statement would resemble the following:

```
SELECT employees.*, roles.* FROM employees
OUTER JOIN roles ON employees.roleid = roles.roleid
```

In this case, all data is returned, even if no role is present for Wilbur and Agnes.