COURSEWARE

Joins

Contents

Overview

Joins are used to combine different tables together based on common data values, realising the necessity for relationships between tables.

The most common usage is with keys, matching a `PRIMARY KEY` in one table to a `FOREIGN KEY` in another table, but that's not a strict requirement; you can match any two fields.

Joins

Let's look at each join in action for the two tables below:

`CUSTOMERS`

customer_id	forename	address
1	Jeff	Dallas
2	Cyrus	Midland
3	William	Pecos
4	Jenny	Montana

`ORDERS`

order_id	fk_customer_id	total
1	2	34.99
2	1	35.00
3	3	32.49
4	6	66.66

Inner Joins

An *inner join* is the default type of `JOIN` which MySQL uses.

It is used to combine tables according to data which is present in both tables.

The syntax for a standard inner `JOIN` consisting of two fields looks like this:

```
SELECT *
FROM table1 t1
JOIN table2 t2 ON t1.field1=t2.field2;
```

In this example, we've given each table an *alias* for brevity and ease of reading - table1 is called `t1`, and table2 is called `t2`.

`JOIN` will only work if it's used as part of a `SELECT` statement, because it specifies which fields to find from tables.

Our `FROM` statement works as usual, with the exception of our table-name aliasing.

We then use the `JOIN` keyword and specify the table in which we're trying to find matching fields, followed by its own alias.

Finally, we use the `ON` keyword to specify the fields which we are trying to match, mapped and identified through the table aliases.

A more realistic example would be to join on keys - if we were trying to find all customer data for an order, we would join the `PRIMARY KEY` in the customers table with the `FOREIGN KEY` in the orders table:

```
SELECT o.order_id, c.customer_id, c.forename, c.address
FROM customers c
JOIN orders o ON c.customer_id=o.fk_customer_id;
```

Naming the `FOREIGN KEY` columns with the `fk` prefix may help to clarify your process.

Let's look at what this statement generates:

order_id	customer_id	forename	address
1	2	Cyrus	Midland
2	1	Jeff	Dallas
3	3	William	Pecos

We can now view the details for each customer that made an order - with no `NULL` values present.

Because Jenny didn't make an order, they don't show up here - outer joins only account for records with *complete data between both tables selected*.

Outer Joins

An *outer join* works a little bit differently.

Not only does it return a set of records that include what an inner join would return, but it also includes other records for which **no corresponding match is found** in the other table.

There are two types of outer join - *left* and *right*.

Left Outer Joins

Left outer joins will produce a result that has a row for every row in the 'left-hand' table (the one you write first in the query), regardless of whether there is a match in the 'right-hand' table, and fill any other field with NULL values.

```
SELECT o.order_id, c.customer_id, c.forename, c.address, o.total
FROM customers c
LEFT OUTER JOIN orders o ON c.customer_id=o.fk_customer_id;
```

This will display all data from the customers table and, if there are no matches in the order table, the value `NULL` will be displayed.

The results we'd get if we ran our *left outer join* query would look like this:

RESULTS

order_id	customer_id	forename	address	total
1	2	Cyrus	Midland	34.99
2	1	Jeff	Dallas	35.00
3	3	William	Pecos	32.49
NULL	4	Jenny	Montana	NULL

Jenny now shows up, even though she didn't make any orders - so the columns we get from the *orders* table are designated `NULL`.

Right Outer Joins

Right outer joins will produce a result that has a row for every row in the 'right' table (the one you write first in the query) regardless of whether there is a match in the 'left' table, and fill any other field with NULL values.

```
SELECT o.order_id, c.customer_id, c.forename, c.address, o.total
FROM customers c
RIGHT OUTER JOIN orders o ON c.customer_id=o.fk_customer_id;
```

This will display all data from the order table and if there are no matches in the customer table, the value NULL will be displayed.

The results we'd get if we ran our *right outer join* query would look like this:

RESULTS

order_id	customer_id	forename	address	total
1	2	Cyrus	Midland	34.99
2	1	Jeff	Dallas	35.00
3	3	William	Pecos	32.49
4	NULL	NULL	NULL	66.66

Now, the order without any customer information attached to it shows up - so the columns we get from the *customers* table are designated `NULL`.

Tutorial

There is no tutorial for this module.

Exercises

There are no exercises for this module.