6 Examples of Joins

Examples of SQL queries that use joins to help you understand the concepts of joining tables

Ben Brumm

www.databasestar.com

Welcome!

Learning how to use joins in SQL is a helpful concept.

You've read my article on Joins that explains the different types of joins with some examples.

But what if you want to write more complicated queries?

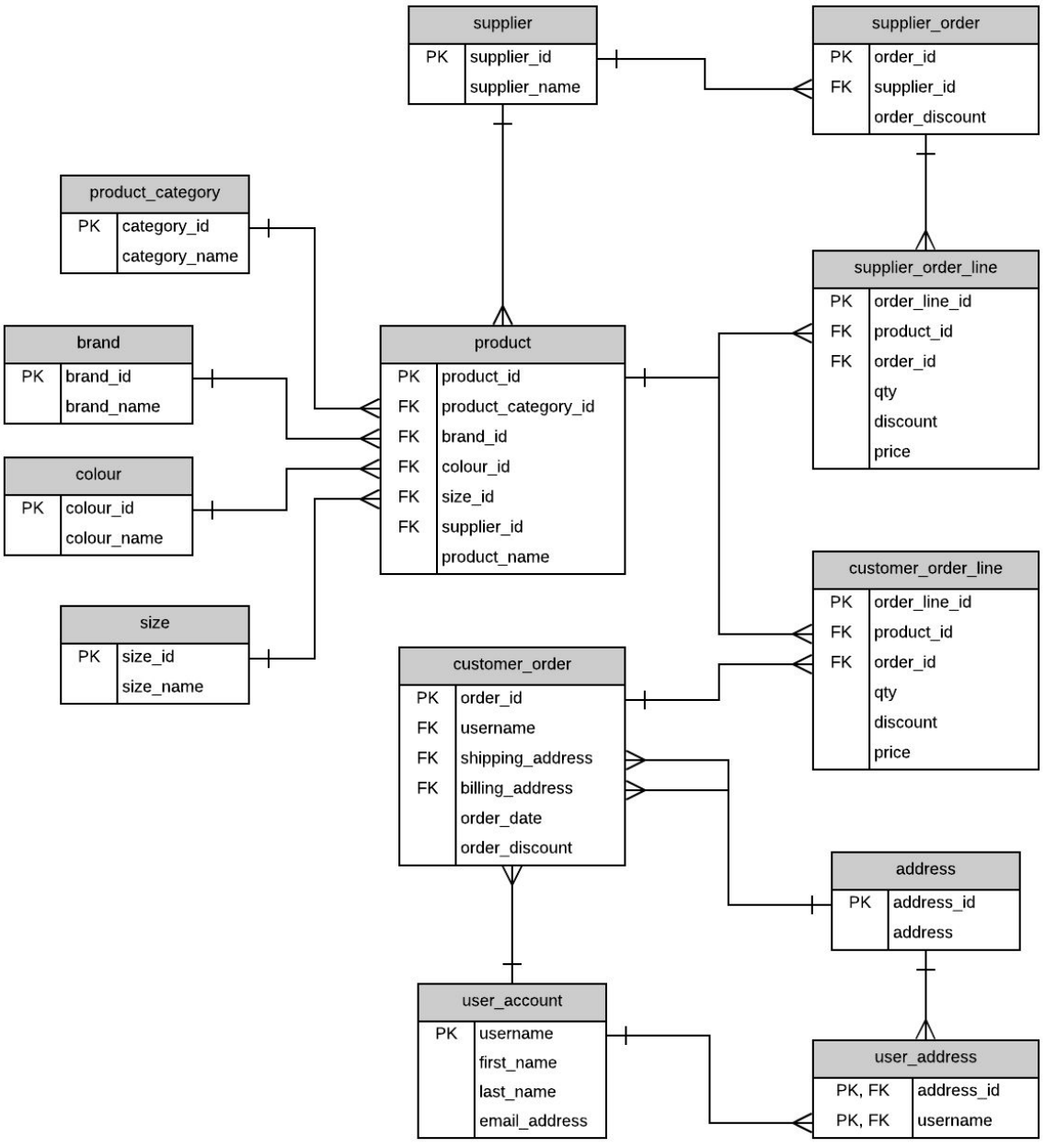
In our jobs, we're often doing more than just INNER JOINs between two tables.

In this guide, I'll share a few examples of more complicated joins, to help you understand how many tables can be joined together.

Sample Database

This is the sample database we're using. It's an online store, which contains data about customers from the website (user_account table), their orders and order_lines, the products, separate tables for each of their attributes (e.g. colour and size), supplier information, and orders to those suppliers.

Here's the ERD:



Now, let's take a look at some examples of queries that use joins.

You'll notice that I've used table aliases in all of the queries. While table aliases are not required, I've used them because they are good practice, and they make it easier to write queries.

Example 1: Orders and Products for a Customer

For this example, we want to:

Find all orders and products and order details for a customer.

To do this, our query would look like this:

```
SELECT
co.order_id,
co.username,
co.shipping_address,
co.billing_address,
co.order_date,
co.order_discount,
col.order_line_id,
col.qty,
col.discount,
col.price,
p.product_id,
p.product_name
FROM customer_order co
INNER JOIN customer_order_line col ON co.order_id = col.order_id
INNER JOIN product p ON col.product_id = p.product_id
WHERE co.username = 'jsmith';
```

I've selected most fields from the three tables mentioned. This will give an output that shows a lot of information for the orders.

An INNER JOIN has been used in all cases, because we don't want to see orders without order lines, or order lines without products.

Also, I've used a WHERE clause of "jsmith" but the idea is that any username can be used here.

Example 2: User's Addresses

For this example, we want to:

Find the user's name and all addresses that a user has for their account.

This can be done with a combination of the user_account, user_address, and address tables.

```
SELECT
u.username,
u.first_name,
u.last_name,
a.address_id,
a.address
FROM user_account u
INNER JOIN user_address ua ON u.username = ua.username
INNER JOIN address a ON ua.address_id = a.address_id
WHERE u.username = 'jsmith';
```

We're selecting data from the user_account and address table. We don't need to view any data from the user_address table, because that's just a joining table and both of the fields there are stored elsewhere.

Also, as above, we limiting on a username of "jsmith", but this can be changed to whatever username is needed.

Example 3: Products and Attributes

We want a query that will:

Find all products and all of their attributes (category, brand, colour, supplier, size).

Our query would look like this:

```
SELECT
p.product_id,
p.product_name,
pc.category_name,
b.brand_name,
c.colour_name,
sz.size_name,
su.supplier_name
FROM product p
INNER JOIN product_category pc ON p.product_category_id = pc.category_id
INNER JOIN brand b ON p.brand_id = b.brand_id
INNER JOIN colour c ON p.colour_id = c.colour_id
INNER JOIN size sz ON p.size_id = sz.size_id
INNER JOIN supplier su ON p.supplier_id = su.supplier_id;
```

We've displayed the name from each table and not the ID. But we have used the ID for each table's join.

There are joins from the product table to five other tables, making six tables in this query in total.

Example 4: Customer Orders and Products

In this example, we want to:

Find all customers, any orders they have placed (if there are any), and all of the products on those orders.

Our query would look like this:

```
SELECT
u.username,
u.first_name,
u.last_name,
co.order_id,
co.order_date,
co.order_discount,
col.order_line_id,
col.qty,
col.discount,
col.price,
p.product_id,
p.product_name
FROM user_account u
LEFT JOIN customer_order co ON u.username = co.username
INNER JOIN customer_order_line col ON co.order_id = col.order_id
INNER JOIN product p ON col.product_id = p.product_id;
```

This is the first query we've used a LEFT JOIN on.

The reason for this is that we wanted to see all customers, and any orders they have placed (if there are any).

We want to see customers even if they don't have any orders. If we used an INNER JOIN, we would be eliminating any customers that don't have orders.

So, we use a LEFT JOIN from user_account to customer_order, which allows us to keep showing the user_accounts that don't have matching records in the customer_order table. The columns from the customer_order table will show as NULL for the rows that don't have orders.

We have also done an INNER JOIN for the other two tables - customer_order_line and product. We want to see all order lines and products, where an order exists. There's no rule saying we need to have the same join type for each table.

Example 5: Products, Attributes, and Orders

In this example, we want to:

Find all products, all of their attributes, and any orders they have placed (if there are any).

Our query would be:

```
SELECT
p.product_id,
p.product_name,
pc.category_name,
b.brand_name,
c.colour_name,
sz.size_name,
su.supplier_name,
co.order_id,
co.order_date,
col.order_line_id,
col.qty
FROM product p
INNER JOIN product_category pc ON p.product_category_id = pc.category_id
INNER JOIN brand b ON p.brand_id = b.brand_id
INNER JOIN colour c ON p.colour_id = c.colour_id
INNER JOIN size sz ON p.size_id = sz.size_id
INNER JOIN supplier su ON p.supplier_id = su.supplier_id
LEFT JOIN customer_order_line col ON p.product_id = col.product_id
INNER JOIN customer_order co ON col.order_id = co.order_id;
```

We have selected from the product and attribute tables, using INNER JOINs, in the same way as we did earlier.

However, we're now joining to the customer_order_line using a LEFT JOIN. This is because we want to show all products, even if there are no orders. The products are joined to orders through the customer_order_line table, so that's where we do the left join.

We have done an INNER JOIN to the customer_order table, as we don't want to show orders without order lines.

Example 6: Products, Suppliers, and Supplier Orders

In this example, we want to:

Find all products, their suppliers, and any orders places with those suppliers and their order lines

Our query would look like this:

```
SELECT
p.product_id,
p.product_name,
s.supplier_id,
s.supplier_name,
so.order_id,
so.order_discount,
sol.order_line_id,
sol.qty,
sol.discount,
sol.price
FROM product p
INNER JOIN supplier s ON p.supplier_id = s.supplier_id
LEFT JOIN supplier_order so ON so.supplier_id = s.supplier_id
INNER JOIN supplier_order_line sol ON so.order_id = sol.order_id;
```

This shows all products and their suppliers. It uses a LEFT JOIN on the supplier_order table, as we want to see all products and suppliers even if there is no order for them.

Even though the supplier_order_line table is joined to the product table, we haven't joined those tables there. If we did, we would only see order lines for the product we're looking at. We wouldn't see order lines for the entire order.

So, as you can see, we can use different types of joins to get data from a database in whatever structure we like. Having a normalised database, with data in different tables, is easy to maintain. We can use joins to get whatever data we need for reporting or applications.

If you have any questions on joins, let me know at ben@databasestar.com

Thanks!

Ben