JOIN USING to Link By Column

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores inner joins with USING to join tables in two parts:

1. Inner JOIN with USING
2. Adding Complexity

1. Inner JOIN with USING

As we have discussed, natural joins are not always ideal, because they will join all common columns between the two tables. Although in some cases this may not be an issue, there can be common columns for a variety of purposes.

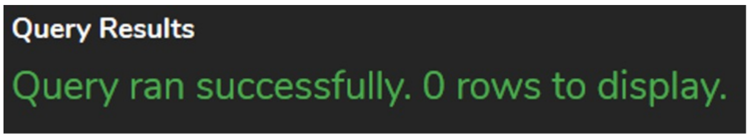
Let's take a look at the product and category tables that we created in the prior tutorial, and which had an issue with the natural join:

```
CREATE TABLE category ( category_id serial PRIMARY KEY, name VARCHAR (100) NOT NULL );

CREATE TABLE product ( product_id serial PRIMARY KEY, name VARCHAR (100) NOT NULL, category_id INT NOT NULL, FOREIGN KEY (category_id) REFERENCES category (category_id) )

INSERT INTO category (name)
VALUES ('Game'), ('Movie'), ('CD');

INSERT INTO product (name, category_id)
VALUES ('Call of Duty', 1), ('Final Fantasy', 1), ('Wizard of Oz', 2), ('Jaws', 2), ('Great Hits', 3), ('Journey', 3);
```

When we tried to run a natural join:

```
SELECT *
FROM product
NATURAL JOIN category;
```

We got the following result:



```
Query Results
Query ran successfully. 0 rows to display.
```

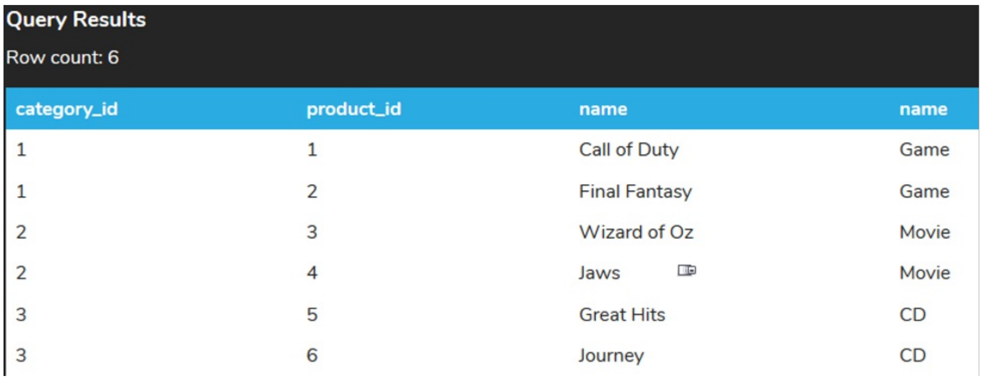
This was due to the fact that the tables had two common attributes. However, the name columns did not have values that matched between the two tables. The JOIN and USING clauses will help with this, and allow us to pick which common attribute to join.

The syntax for the command looks like this:

```
SELECT <columnlist>
FROM <table1>
INNER JOIN <table2> USING (<commonattribute>);
```

In our set of tables, we can use the JOIN and USING clauses to focus on category_id:

```
SELECT *
FROM product
INNER JOIN category USING (category_id);
```

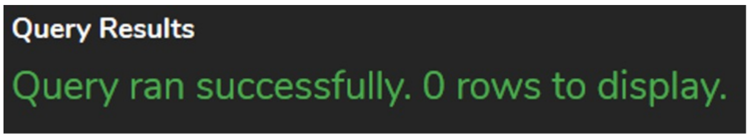

Query Results

Row count: 6

category_id	product_id	name	name
1	1	Call of Duty	Game
1	2	Final Fantasy	Game
2	3	Wizard of Oz	Movie
2	4	Jaws	Movie
3	5	Great Hits	CD
3	6	Journey	CD

Since name also exists in both tables, we can also use name:

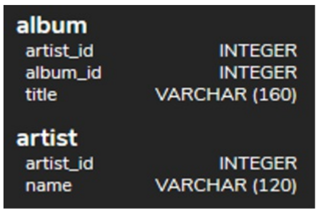
```
SELECT *
FROM product
INNER JOIN category USING (name);
```


Query Results

Query ran successfully. 0 rows to display.

However, again, since none of the names between the tables match, no rows are returned.

Let us go back to some of our other tables, like the album and artist table.


album

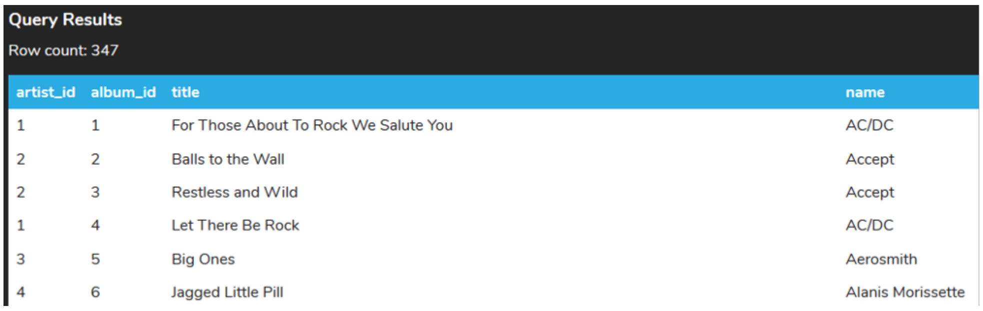
artist_id	INTEGER
album_id	INTEGER
title	VARCHAR (160)

artist

artist_id	INTEGER
name	VARCHAR (120)

Right now, we have only focused on each individual table. We don't know which album has which artist unless we look up the artist_id. However, since the artist_id exists in both tables, we can use the JOIN and USING clause to join the tables on the artist_id:

```
SELECT *
FROM album
INNER JOIN artist USING (artist_id);
```


Query Results

Row count: 347

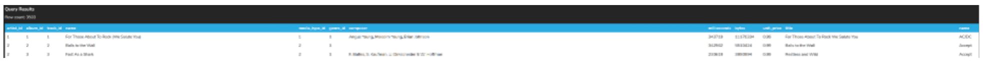
artist_id	album_id	title	name
1	1	For Those About To Rock We Salute You	AC/DC
2	2	Balls to the Wall	Accept
2	3	Restless and Wild	Accept
1	4	Let There Be Rock	AC/DC
3	5	Big Ones	Aerosmith
4	6	Jagged Little Pill	Alanis Morissette

Now our data is starting to make a bit more sense, as we join it together.

2. Adding Complexity

We can add more than just two tables together by adding additional INNER JOIN statements with USING. If we take the example of artist and album, we can also identify the tracks on each:

```
SELECT *
FROM track
INNER JOIN album USING (album_id)
INNER JOIN artist USING (artist_id);
```


Query Results

Row count: 3503

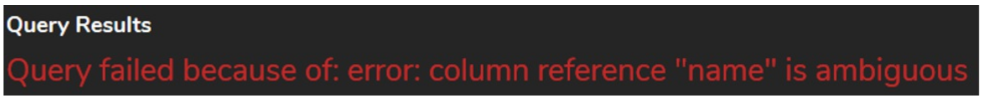
artist_id	album_id	track_id	name	media_type_id	genre_id	composer	milliseconds	bytes	unit_price	title	name
1	1	1	For Those About To Rock (We Salute You)	1	1	Angus Young, Malcolm Young, Brian Johnson	343719	11170334	0.99	For Those About To Rock We Salute You	AC/DC
2	2	2	Balls to the Wall	2	1		342562	5510424	0.99	Balls to the Wall	Accept
2	3	3	Fast As a Shark	2	1	F. Baltes, S. Kaufman, U. Dirkscneider & W. Hoffman	230619	3990994	0.99	Restless and Wild	Accept

As you can see, by using the * as we start to join more tables, we may have too many columns being returned. We can specify in the SELECT clause which columns should be returned. It is a best practice to use the format <tablename>.<columnname> when we list the columns. Otherwise, if a column name exists in more than one table, the database does not know which column you want to display and returns an ambiguous error. For example, "name" exists in the track and artist table. If we try to simply include the name column in the

SELECT clause:

```
SELECT name
FROM track
INNER JOIN album USING (album_id)
INNER JOIN artist USING (artist_id);
```

We will get the following error message:



Query Results

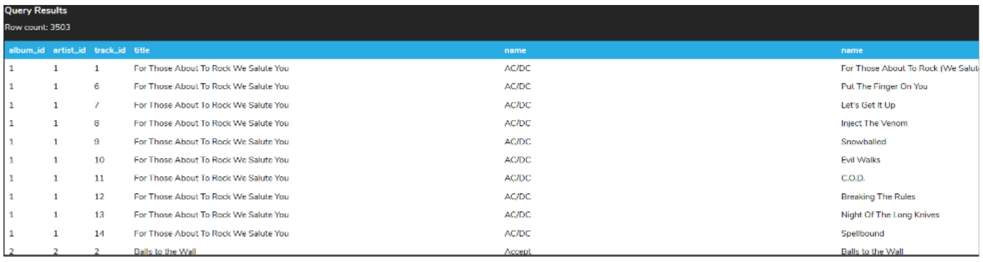
```
Query failed because of: error: column reference "name" is ambiguous
```

Note with an INNER JOIN, you can add in the artist_id, album_id, and track_id as the result set only has one item. However, this is not a good practice.

```
SELECT album_id, artist_id, track_id, album.title, artist.name, track.name
FROM track
INNER JOIN album USING (album_id)
INNER JOIN artist USING (artist_id)
ORDER BY album_id, artist_id, track_id;
```

It would be a better practice to use the same labeling as the following:

```
SELECT album.album_id, artist.artist_id, track.track_id, album.title, artist.name, track.name
FROM track
INNER JOIN album USING (album_id)
INNER JOIN artist USING (artist_id)
ORDER BY album.album_id, artist.artist_id, track.track_id;
```



Query Results

Row count: 3503

album_id	artist_id	track_id	title	name	name
1	1	1	For Those About To Rock We Salute You	AC/DC	For Those About To Rock (We Salut
1	1	6	For Those About To Rock We Salute You	AC/DC	Put The Finger On You
1	1	7	For Those About To Rock We Salute You	AC/DC	Let's Get It Up
1	1	8	For Those About To Rock We Salute You	AC/DC	Inject The Venom
1	1	9	For Those About To Rock We Salute You	AC/DC	Snowballed
1	1	10	For Those About To Rock We Salute You	AC/DC	Evil Walks
1	1	11	For Those About To Rock We Salute You	AC/DC	C.O.D.
1	1	12	For Those About To Rock We Salute You	AC/DC	Breaking The Rules
1	1	13	For Those About To Rock We Salute You	AC/DC	Night Of The Long Knives
1	1	14	For Those About To Rock We Salute You	AC/DC	Spellbound
2	2	2	Balls to the Wall	Accept	Balls to the Wall

Video Transcription

[MUSIC PLAYING] In this tutorial, we're going to explore utilizing an inner join with the USING keyword.

So in this case here, we're going to take a look at the two tables, category and product. As part of the prior task, we explored it when utilizing the same category_id as a reference. However, in this case, there's a small change with the two way tables having name on both tables as well.

So in this case here, we've already inserted the data. However, the problem is, when it comes to a natural join, there's two different columns that match up in terms of the name of the columns. However, the data doesn't match up. If we try to run a natural join, it'll try to link on both the category_id as well as the name. Hence, no different rows are going to be displayed in this case.

What we want to do is actually create an inner join instead. And so as part of that, we can make a quick change, in this case here, from a NATURAL to the word INNER, and JOIN with the second name of the table. And then we're going to have the keyword USING and the specific column that we want, which in this case here is going to be in brackets with category_id.

And then we can go ahead and run that. So in this case here, the results are going to be displayed correctly, being that we've actually identified the specific column that we want the two tables to link up.

[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

The INNER JOIN with USING joins tables together using common attributes that are specified.

Source: Authored by Vincent Tran