JOINs That Specify Which Columns to Use

by Sophia

WHAT'S COVERED

In this lesson, you will explore the INNER JOIN clause with USING to join tables, in two parts. Specifically, this lesson will cover:

1. INNER JOIN With USING
2. INNER JOIN USING Examples

1. INNER JOIN With USING

An INNER JOIN in SQL retrieves data from two or more related tables based on a specified condition. A USING clause is often used in conjunction with INNER JOIN to simplify the JOIN condition by specifying which column or columns should be used for the JOIN. Avoiding repetitive column names in the result set helps streamline the query syntax and reduce redundancy.

If the INNER JOIN clause is used with the USING clause, the database engine matches rows where the specified columns have equal values. This process eliminates nonmatching rows from the result set by including only the rows that match the condition. Although the USING clause can simplify the JOIN process, it is important to note that it restricts joining to columns with identical names. When columns have the same name but represent different data, or when joining conditions are more complex, using the ON clause might be a better approach. You will learn about the ON clause in the next lesson.

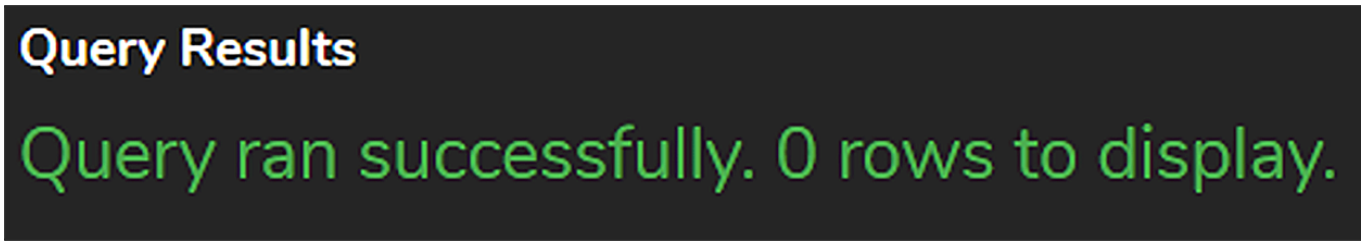
Let's take a look at the product and category tables that we created in the prior lesson, which had an issue with the NATURAL JOIN:

```
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) REFI
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);
```

Consider when we tried to run a NATURAL JOIN:

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

We got the following result:



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 enable 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

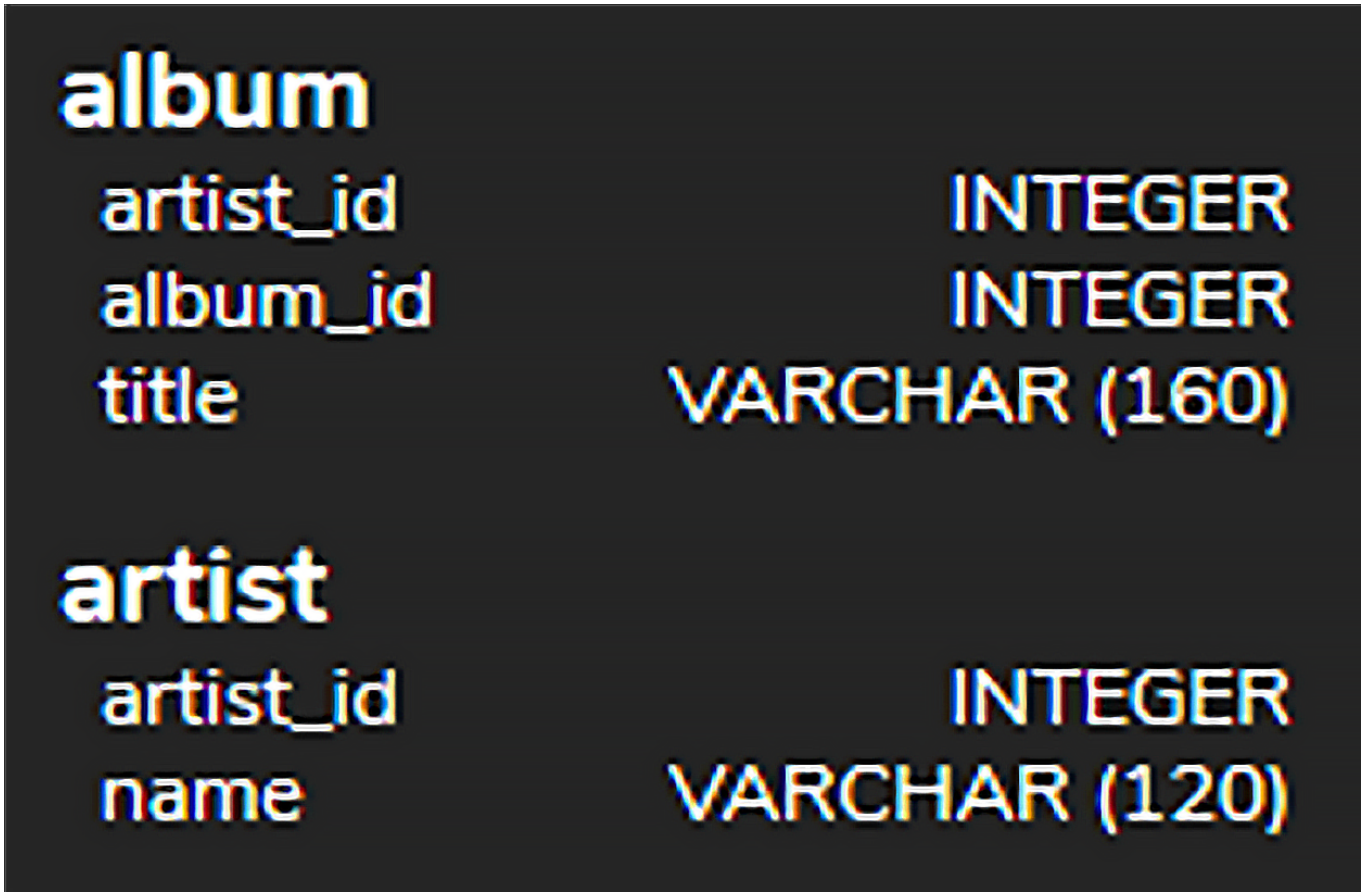
category_id	product_id	name	name
1	1	Call of Duty	Game
1	2	Final Fantasy	Game
2	3	Wizard of Oz	Movie
2	4	Jaws	Movie
3	5	Great Hits	CD
3	6	Journey	CD

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

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

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

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



So far, 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. INNER JOIN USING Examples

We can join more than just two tables 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 multiple tables, 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. Consider if we try to simply include the name column in the SELECT clause:

```
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:

If a particular column name appears in only one of the tables involved, you can reference it by its name only, omitting the table name. For example, you could write the following statement where only certain columns have the table name as a prefix:

```
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;
```

However, this is not a good practice because at some point the table(s) or the query may be changed so that there are duplicate names. It is a better practice to use consistent naming, with all the table names explicitly specified, like this:

```
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;
```

WATCH

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

In this lesson, you learned that using an **INNER JOIN**, you can combine data from two or more related tables based on a defined condition. The **USING** clause enables you to specify a single column that should be used for the INNER JOIN operation when you use INNER JOIN. It simplifies the query syntax by removing the need to specify redundant column names.

In an INNER JOIN with USING, rows from joined tables are matched based on the specified column. The column name and data type must be the same in both tables. As a result, only rows with matching values in the specified column will be included, effectively eliminating nonmatching records. Using the USING clause offers a convenient way to perform INNER JOINs, but it's important that the column being used has consistent and meaningful data across tables, especially when the complexity increases. For accurate and intended results, it's imperative to pay attention to the column characteristics to simplify queries and make them easier to read, especially when dealing with tables with common attributes.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR **TERMS OF USE**.