VIEW to Simplify Queries

by Sophia

WHAT'S COVERED

This tutorial explores using views to join multiple table data to simplify queries in two parts:

1. Combining Data
2. Complex Query Example

1. Combining Data

Beyond just querying from a single table, you can also use views to combine data from multiple tables. For example, seeing the support_rep_id may not be extremely useful in an organization unless you know who that value belongs to. Instead, you could include the name of the support rep, similar to the following:

```
CREATE VIEW customer_contact
AS
SELECT customer.*, employee.first_name as support_first_name, employee.last_name as support_last_name
FROM customer, employee
WHERE customer.support_rep_id = employee.employee_id;
```

If we queried the customer_contact view, it would look like the following:

```
SELECT *
FROM customer_contact;
```


Query Results
Row count: 59

customer_id	first_name	last_name	company	address	city	state	country	postal_code	phone	fax	email	support_rep_id	support_first_name	support_last_name
1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555	+55 (12) 3923-5566	luisg@embraer.com.br	3	Jane	Peacock
2	Leonie	Köhler		Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	+49 0711 2842222		leonekohler@surfeu.de	5	Steve	Johnson
3	François	Tremblay		1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711		ftremblay@gmail.com	3	Jane	Peacock

We could further add to the query to include the necessary columns:

```
SELECT first_name, last_name, support_first_name, support_last_name
FROM customer_contact;
```

Query Results

Row count: 59

first_name	last_name	support_first_name	support_last_name
Luís	Gonçalves	Jane	Peacock
Leonie	Köhler	Steve	Johnson
François	Tremblay	Jane	Peacock

If we wanted to write this out using tables, we would have to do the following each time:

```
SELECT customer.first_name, customer.last_name, employee.first_name as support_first_name, employee.last_name as support_last_name
FROM customer, employee
```

WHERE customer.support_rep_id = employee.employee_id;

2. Complex Query Example

Let's take a look at a more complex query that uses more than 2 tables. So far, when we query the tables for the tracks, we are looking only at the id values. You may also want to get the artist's name, album title, and track name at the same time. Creating a view for this purpose can simplify this process:

```
CREATE VIEW artist_album_track
AS
SELECT artist.name as artist_name, album.title as album_title, track.name as track_name
FROM artist
INNER JOIN album ON artist.artist_id = album.artist_id
INNER JOIN track ON album.album_id = track.album_id;
```

Rather than querying the tables to get that list:

```
SELECT artist.name as artist_name, album.title as album_title, track.name as track_name
FROM artist
INNER JOIN album ON artist.artist_id = album.artist_id
INNER JOIN track ON album.album_id = track.album_id;
```

We can simply query the view directly, like this:

```
SELECT *
FROM artist_album_track;
```

Query Results

Row count: 3503

artist_name	album_title	track_name
AC/DC	For Those About To Rock We Salute You	For Those About To Rock (We Salute You)
Accept	Balls to the Wall	Balls to the Wall
Accept	Restless and Wild	Fast As a Shark
Accept	Restless and Wild	Restless and Wild
Accept	Restless and Wild	Princess of the Dawn

If we wanted to add some filters into our SELECT statement, such as only listing the rows that belong to AC/DC, instead of doing this:

```
SELECT artist.name as artist_name, album.title as album_title, track.name as track_name
FROM artist
INNER JOIN album ON artist.artist_id = album.artist_id
INNER JOIN track ON album.album_id = track.album_id
WHERE artist.name = 'AC/DC';
```

Query Results

Row count: 18

artist_name	album_title	track_name
AC/DC	For Those About To Rock We Salute You	For Those About To Rock (We Salute You)
AC/DC	For Those About To Rock We Salute You	Put The Finger On You
AC/DC	For Those About To Rock We Salute You	Let's Get It Up
AC/DC	For Those About To Rock We Salute You	Inject The Venom

We would query the view like this:

```
SELECT *
FROM artist_album_track
WHERE artist_name = 'AC/DC';
```

Query Results

Row count: 18

artist_name	album_title	track_name
AC/DC	For Those About To Rock We Salute You	For Those About To Rock (We Salute You)
AC/DC	For Those About To Rock We Salute You	Put The Finger On You
AC/DC	For Those About To Rock We Salute You	Let's Get It Up
AC/DC	For Those About To Rock We Salute You	Inject The Venom

The second option greatly simplifies the query without having to join each of the tables together.

Video Transcription

[MUSIC PLAYING] We can also create views to be able to join different tables together based on common data in this case here. For example, in our scenario, we might want to get the artist's name and the album title as well as the track's name. Without joining different tables together, we would only be left with the actual IDs within the tables.

So in this case here, we'll simplify things by creating a view that joins the album table, the artist table, and the track table together based on their primary and foreign keys, and then only display the artist's name, the album title, and the track name.

So in normal cases, if we wanted to query and get those items, we'd have to write this entire query to be able to display all 3,500 rows. In this case here, we're going to go ahead and create this view. So once we've actually created the view, if we wanted to query from it, we would just simply have this SELECT statement, and from this, view name.

From this view a name, you can add on additional filters that you would typically have in this case as well. There's an additional track name there. We can add in that WHERE clause utilizing the artist's name, for example, where it's equal to Accept. If we run this, we'll be able to quickly get the results without having to write the entire statement to join all the different tables together.

[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

Views allow us to join multiple table data to simplify queries.

Source: Authored by Vincent Tran