CREATE OR REPLACE VIEW to Update Views

by Sophia

WHAT'S COVERED

This lesson explores making changes to views, in two parts. Specifically, this lesson will cover:

1. Introduction

If a view needs to be changed, you have two options.

First, you can use a command that replaces or alters the existing view statement. This method ensures atomicity: Either the view is replaced with the new definition, or it remains unchanged. There is no point at which the view doesn't exist. It also preserves the existing view's dependencies, as long as those dependencies don't rely on columns that have been changed or removed. However, different DBMSs have differing commands and syntax for doing this, and in some systems, the changes you can make are limited; if you need to make big structural changes, you may have to delete and recreate.

In PostgreSQL, you use CREATE OR REPLACE VIEW to make changes to the view's structure, and you use ALTER VIEW to make changes to its non-content properties such as its name and its security permissions.

KEY CONCEPT

You cannot use CREATE OR REPLACE VIEW to drop columns from a view; you just drop and recreate the view if you need that.

Second, you can DROP (delete) and recreate the view. This approach is simple, and it provides complete control over the view's definition and structure. You will learn about that method in the next lesson.

2. Replacing or Altering a View

Some DBMSs (including PostgreSQL, MySQL, and MariaDB) have a CREATE OR REPLACE VIEW statement that you can use to substitute a new view for an old one of the same name. Here's the syntax for doing that:

```
CREATE OR REPLACE VIEW my_view AS
SELECT column1, column2, ..
FROM some_table
WHERE condition;
```

There are limitations to its use, though, and those limitations can vary depending on the DBMS.

HINT

Some DBMSs call this operation CREATE OR ALTER VIEW. SQL Server is one example.

PostgreSQL also has an ALTER VIEW statement, but it is limited. It does not change the content of the underlying query; it only changes metadata about the view, such as its name, owner, or security policies. The basic syntax is:

```
ALTER VIEW view_name OPTIONS (view_options);
```

Let's work through an example that demonstrates how to use both the CREATE OR REPLACE VIEW and the ALTER VIEW statements.

Let's create a view that has the invoice_id, total, and first and last name of the customer:

```
CREATE VIEW invoice_information
AS
SELECT invoice_id, total, first_name, last_name
FROM invoice
INNER JOIN customer ON invoice.customer_id = customer.customer_id;
```

To query the view, we can run the following SELECT statement:

```
SELECT *
FROM invoice_information;
```

Query Results

Row count: 412

invoice_id	total	first_name	last_name
1	2	Leonie	Köhler
2	4	Bjørn	Hansen
3	6	Daan	Peeters
4	9	Mark	Philips
5	14	John	Gordon
6	1	Fynn	Zimmermann
7	2	Niklas	Schröder
8	2	Dominique	Lefebvre

If we wanted to have the employee's first and last name instead of the customer's first and last name, we could drop the view and then recreate it. However, if there were objects that depended on this view, we would have to drop those as well. Using the CREATE OR REPLACE VIEW allows us to recreate it without getting rid of the dependent objects.

Let us make the change in the view:

```
CREATE OR REPLACE VIEW invoice_information
AS
SELECT invoice_id, total, employee.first_name, employee.last_name
FROM invoice
INNER JOIN customer ON invoice.customer_id = customer.customer_id
INNER JOIN employee ON customer.support_rep_id = employee.employee_id;
```

Even though the names of the columns are the same and are in the same order, we will get this error:

Query Results

Query failed because of: error: cannot change data type of view column "first_name" from character varying(40) to character varying(20)

This is because the first_name in the employee table is 20 characters, so the data size does not match. In this case, we could resize the employee table's first_name column to 40 characters, as well as the last_name:

```
ALTER TABLE employee
ALTER COLUMN first_name TYPE VARCHAR (40);
ALTER TABLE employee
ALTER COLUMN last_name TYPE VARCHAR (40);
```

Now, let's attempt to make the modification again:

```
CREATE OR REPLACE VIEW invoice_information
AS
SELECT invoice_id, total, employee.first_name, employee.last_name
FROM invoice
INNER JOIN customer ON invoice.customer_id = customer.customer_id
INNER JOIN employee ON customer.support_rep_id = employee.employee_id;
```

We are successful, as the data type, column names, and order all match:

```
SELECT *
FROM invoice_information
```

Query Results

Row count: 412

invoice_id	total	first_name	last_name
1	2	Steve	Johnson
2	4	Margaret	Park
3	6	Margaret	Park
4	9	Steve	Johnson
5	14	Margaret	Park
6	1	Jane	Peacock
7	2	Jane	Peacock
8	2	Margaret	Park

Since the data has changed, we may want to also change the name of the view. We can use the ALTER VIEW statement to rename it:

```
ALTER VIEW invoice_information RENAME TO invoice_employee_info;
```

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.

In the **introduction**, you learned that when you want to make changes to a view, you have two options: **replacing or altering a view**. You can use the CREATE OR REPLACE VIEW statement (to make changes to the underlying query) or the ALTER VIEW statement (to make changes to the view's metadata, such

as its name). Using CREATE OR REPLACE VIEW, the existing query in the view is completely replaced with a new one. You saw some difficulties that might arise due to column size, and how to overcome them. You also learned that another option is to DROP (delete) and recreate a view, a topic that will be covered in the next lesson.

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