

VIEW to Provide a Subset

by Sophia



WHAT'S COVERED

This tutorial explores using views to provide a useful report on a subset of data in two parts:

1. Introduction
2. VIEW Examples

1. Introduction

A view is a named query that allows us to present the data in a database in a different way. When we create a view, we are basically creating a query and then assigning it a name. Then we can query the view for simplicity. These named queries that we create do not store any data directly, unless they are materialized views. Those are special types of views that store the data physically and refresh the data from the base tables that the views are dependent on.

2. VIEW Examples

A view can help simplify the complexity of a query, as we will see in the upcoming tutorials, by allowing us to bypass entering in the complex query each time. In addition, like a table, you can grant permission to users through a view that contains specific data. For example, if you have an employee table that has sensitive personal information like salary, you could create a view that does not include the salary and allow users to query that view rather than the base table.

The statement would look like the following:

```
CREATE VIEW <viewname>
```

```
AS
```

```
SELECT <statement>;
```

For example, if we wanted to only have a list of the employee names and who they report to, we could create a view like:

```
CREATE VIEW employee_report
```

```
AS
```

```
SELECT first_name, last_name, employee_id, reports_to
```

FROM employee;



```
employee_report
reports_to      INTEGER
employee_id     INTEGER
last_name       VARCHAR (20)
first_name      VARCHAR (20)
```

To query from this view, we would create a SELECT statement as if it were querying from a table:

```
SELECT *
FROM employee_report;
```

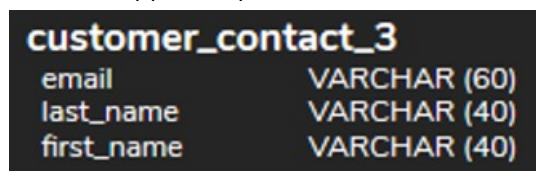


Row count: 8			
first_name	last_name	employee_id	reports_to
Andrew	Adams	1	
Nancy	Edwards	2	1
Jane	Peacock	3	2
Margaret	Park	4	2
Steve	Johnson	5	2
Michael	Mitchell	6	1
Robert	King	7	6
Laura	Callahan	8	6

If this were a query that we ran often, it would simplify having to write out the query each time.

If different employees wanted to query their customer list (through the support_rep_id), we could create a view for each:

```
CREATE VIEW customer_contact_3
AS
SELECT first_name, last_name, email
FROM customer
WHERE support_rep_id = 3;
```



```
customer_contact_3
email      VARCHAR (60)
last_name  VARCHAR (40)
first_name VARCHAR (40)
```

Instead of having to query that entire customer list and filtering out the data each time, the support_id_rep equal to 3 can query the customer_contact_3 view directly, like:

```
SELECT *
FROM customer_contact_3;
```

Otherwise, they would have to enter in the entire SELECT statement each time:

```
SELECT first_name, last_name, email
FROM customer
WHERE support_rep_id = 3;
```

In this case, the view filters the columns (first_name, last_name, and email) as well as the rows (those that have the support_rep_id equal to 3). Since the view does not store any data, if anything changes in the customer table, the updates would be reflected in the view immediately when queried.

Video Transcription

[MUSIC PLAYING] We can create views to be able to create subsets of data to be able to query from. So if there's a query that we run quite often, it might be complex and might have lots of different components. Rather than rewriting the query each time and then querying from that, we can be able to create these views to quickly query the underlying data.

So in this case here going want to go ahead and create a view called customer_contact_3 that queries and limits the customer table data to just the first name, last name, and email. And then we're also going to set up a row filter to only have those that have the support rep ID as equal to 3. So this thing would be geared directly to the support rep id 3. So if you wanted to query from that, it would simply do SELECT star from customer_contact_3. So basically the exact same as if you were querying from a table.

And by doing so we'll be able to query the data and only list out the first name, last name, and email of those that have that support rep id equal to 3. So if it's something that you run quite often creating a view might be simplified being that you'd only have to write the details in this case here, you can also filter this out by just having additional filters on top of on this data as if you are just actually creating the underlying SQL select statements.

[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

Using the VIEW allows us to create a named query to query from to simplify SELECT statements.

Source: Authored by Vincent Tran