VIEW to Provide a Subset

by Sophia

WHAT'S COVERED

This lesson explores using CREATE VIEW to provide a useful report on a subset of data, in two parts. Specifically, this lesson will cover:

1. Introduction to Views

So far in this course, every time you wanted to query a database, you have had to manually type the statement. That's a lot of work! One way to avoid all that typing and make accessing the database easier for people who aren't SQL experts is to create views.

A **view** is a saved query that acts as if it were a table. Users can query the view the same way they query a table. Views can simplify complex queries, abstract underlying data structures, and provide some security.

Here are some of the benefits of using views:

- Abstraction. Views enable you to present data in a user-friendly way. For example, a view can combine data from multiple tables and perform calculations on it without the end user needing to know how that happens.
- Security. By creating views that expose only the relevant data, you can control what data users are allowed to see.
- Reusability. You can use a view to save a query that you run frequently so you don't have to retype it each time.
- Performance. When you query a view, the underlying data has already been processed, potentially reducing the execution time of your queries.
- Simplified access: Views can provide a unified interface to data that is stored in different tables. Instead of directly querying multiple tables, users can query views that consolidate the necessary information.

- Integrity: Views can enforce data integrity by ensuring that certain conditions are always met when accessing the data. For example, a view can filter out invalid or inconsistent data.

KEY CONCEPT

Although views are primarily used for reading data, some database systems support updateable views that allow limited data modifications.

TERM TO KNOW

View

In a database, a view is the result set of a stored query, which can be queried like a persistent database collection object.

2. View Examples

A view can help simplify the complexity of a query, as we will see in the upcoming lessons, by allowing us to bypass entering the complex query each time. In addition, like a table, you can grant users permission through a view containing specific data. For example, if you have an employee table with 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.

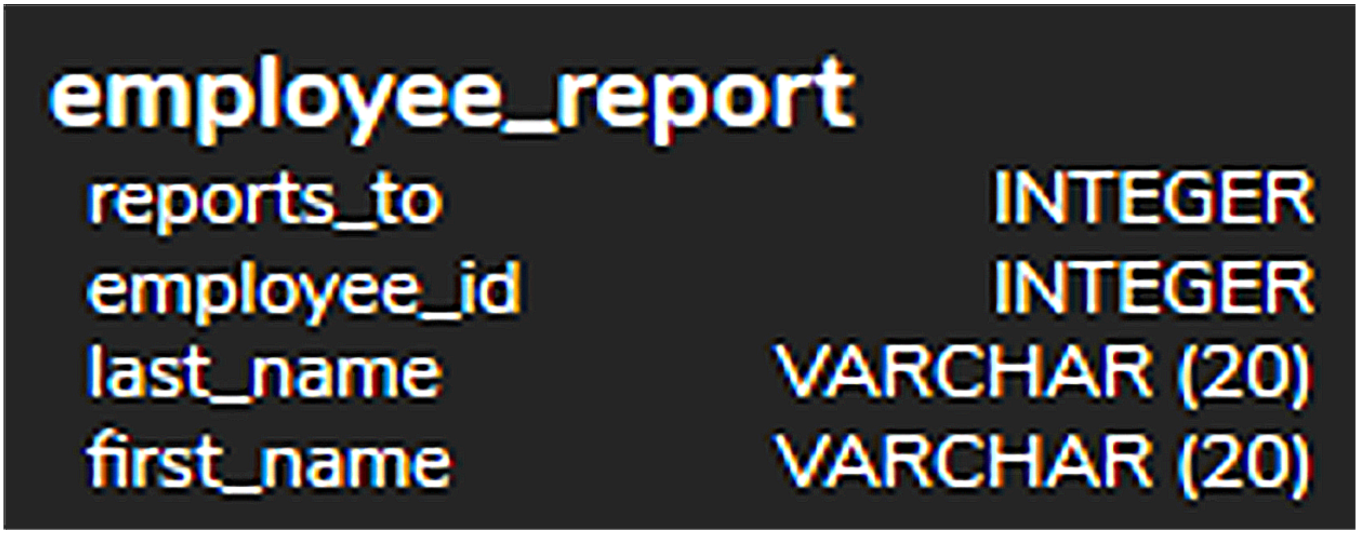
The statement would look like the following:

```
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 this:

```
CREATE VIEW employee_report
AS
SELECT first_name, last_name, employee_id, reports_to
FROM employee;
```

In PostgreSQL, views appear in the navigation pane on the left, like tables do.

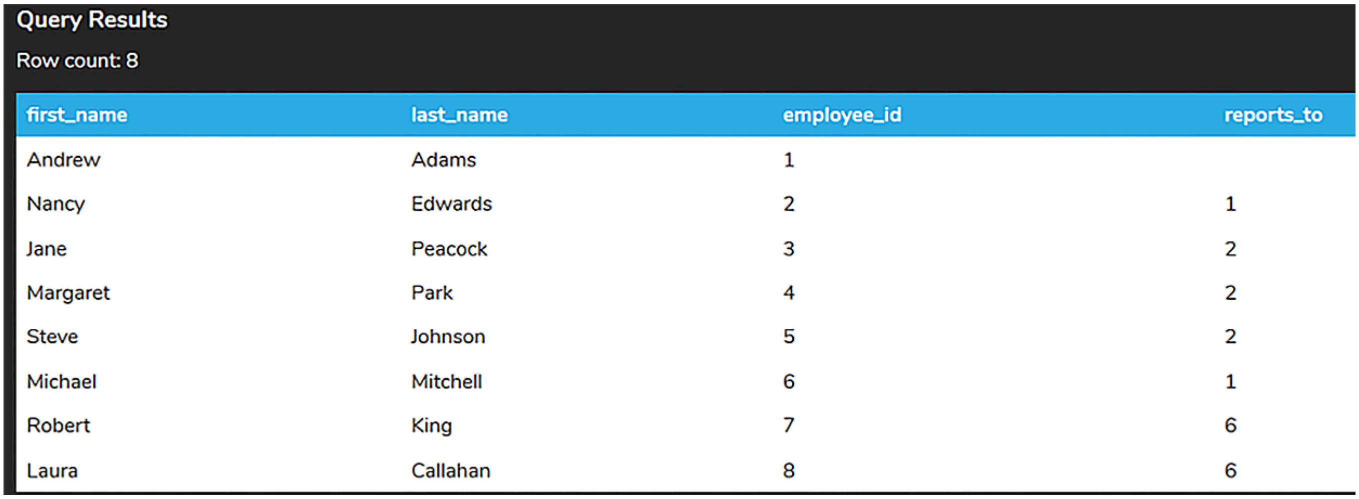


employee_report

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;
```



Query Results

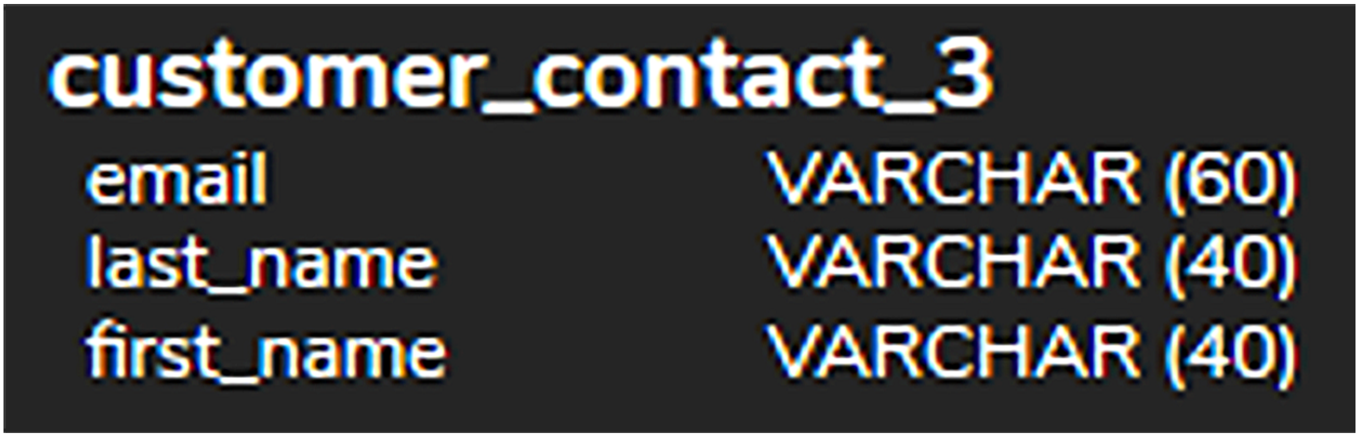
Row count: 8

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 this:

```
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.

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 the **introduction to views** section, you learned that a database view plays a crucial role in enhancing database usability, security, and efficiency. It enables the logical abstraction of complex data structures, providing tailored insights into the underlying data. Users benefit from this abstraction because it simplifies user interactions, makes queries more intuitive, and shields them from the complexities of the database schema. Users can utilize views when they need specific subsets of data or calculations for specific tasks since they can be designed to present only those elements of information that they need. Views reduce the processing load by performing calculations and aggregations in advance. They also reduce the amount of typing needed to execute SQL statements that must be run frequently.

You learned in the **view examples** that the security of data is also significantly enhanced by views. Views enable database administrators to limit user access to sensitive or confidential data by creating views that omit the columns containing the data to be kept private. Users can then query the view instead of the table.

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

TERMS TO KNOW

View

In a database, a view is the result set of a stored query, which can be queried like a persistent database collection object.