Views and Their Applications in Data Analysis

Deepthy A
Data Analyst | Python • SQL • Power BI • Excel | Turning Data into Business Insights

October 14, 2024

In the ever-expanding field of data analysis, the ability to visualize data effectively is paramount. This is where views come into play. A view, in essence, is a saved query that presents a subset of data in a specific format. By creating and utilizing views, data analysts can streamline their workflows, enhance understanding, and derive valuable insights. A **view** is essentially a **virtual table** based on a SQL query that retrieves data from one or more tables. It doesn't store any data itself but provides an abstract layer, allowing users to interact with complex data structures more easily and securely.

What are Views?

A view is a virtual table that is derived from one or more underlying base tables. It **doesn't store actual data** but rather ***presents a customized view of the data*** based on the defined query. This allows analysts to focus on the specific aspects of the data that are relevant to their analysis without having to manipulate the underlying tables directly. Views can encapsulate complex queries and present data in a simplified form.

Syntax to Create a View:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

> This creates a view named view_name that pulls data from table_name based on the specified condition.

Types of Views

When working with **views** in SQL, it's important to understand the distinctions between the two main types of views: **Simple Views** and **Complex Views**. Both types serve different purposes based on how they are constructed and the kinds of queries they encapsulate.

- **Simple View:** Based on a single table and does not include advanced SQL functions, joins, or grouping operations. Simple views are often used to present a cleaner, more concise version of a table, or to hide certain columns from users for privacy or security reasons.

Characteristics of Simple Views:

- **Single table**: Simple views draw data from only one table.
- **No joins**: They do not involve joining multiple tables.
- **No groupings**: Aggregation functions like GROUP BY are not used in simple views.
- **Direct mapping**: Every row in the view corresponds directly to a row in the base table.

Benefits of Simple Views:

- **Easy to maintain**: Since they involve just one table, simple views are straightforward and easier to manage.
- **Security**: By creating simple views, you can limit users' access to only certain columns of the table, effectively enhancing data security.
- **Simplified data access**: Users can access specific parts of the table without needing to filter or format the data manually.

Example of a Simple View:

Let's say we have a table called employees with the following columns: id, name, position, salary, and hire_date. To create a simple view that only displays the name and position of each employee (hiding sensitive information like salary), we could write:

```
CREATE VIEW employee_overview AS
SELECT name, position
FROM employees;
```

> This view, employee_overview, only presents a simplified version of the employee table, showing only the name and position columns.

- **Complex View:** A **Complex View** can be based on **multiple tables** and may include more advanced SQL features such as **joins**, **aggregate functions** (SUM(), COUNT()), and **grouping**. These views allow you to combine data from different sources and present more intricate datasets, making them ideal for complex data analysis and reporting.

Characteristics of Complex Views:

- **Multiple tables**: Complex views often involve more than one table, typically through the use of joins.
- **Joins**: These views may use different types of joins (INNER JOIN, LEFT JOIN, etc.) to retrieve data from related tables.
- **Aggregation**: Complex views can include aggregation functions like COUNT(), SUM(), AVG(), etc., to summarize data.
- **Groupings**: Grouping of data using the GROUP BY clause is common in complex views to organize and aggregate the data.

Benefits of Complex Views:

- **Combining multiple tables**: Complex views allow you to fetch and present data from multiple tables in a single, unified result.
- **Advanced reporting**: You can predefine calculations, aggregations, and joins in a complex view, making reporting easier and more efficient.
- **Reusable queries**: Complex queries that involve multiple tables or conditions can be reused by saving them as views, ensuring consistency across reports or analyses.

Example of a Complex View:

Imagine we have two tables: employees and departments. We want to create a view that lists the name of each employee along with the department they work in. We can use a complex view that involves a join between these two tables:

```
CREATE VIEW employee_department AS
SELECT e.name, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.id;
```

> In this view, employee_department, we are pulling data from both the employees and departments tables, combining them using an INNER JOIN. This is an example of a complex view since it involves multiple tables and a join.

When to Use Simple vs. Complex Views

- **Simple Views** are most appropriate when you need to simplify data access for users or hide certain data from them. They are useful in situations where the data being fetched is straightforward and doesn't require any advanced operations.
- **Complex Views** are ideal when you need to combine data from multiple sources or run aggregate calculations. They allow you to store complex queries involving multiple tables, joins, or even subqueries, which can save time in data analysis and reporting.

Key Benefits of Using Views

- **Simplifying Complex Queries:** When working with databases, it's common to write complex queries that involve multiple tables, joins, and filtering conditions. Instead of repeating the same long query every time, a view can store it for easy reuse. This makes the query more readable and maintainable.

Example:

```
CREATE VIEW employee_info AS
SELECT e.name, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.id;
```

- **Enhancing Data Security:** Views can be used to control access to specific data. For instance, you might want to grant users access only to certain columns in a table, while hiding sensitive data like salaries or personal information.

Example:

```
CREATE VIEW public_employee_info AS
SELECT name, position
FROM employees;
```

This view will restrict users from seeing salary details but allow them to see names and positions.

- **Data Abstraction:** Views offer a level of abstraction by presenting the data in a way that hides the complexity of the underlying database schema. Users interact with simplified representations of the data, which is useful in large systems where the schema might be complex.
- **Optimizing Performance:** While views themselves do not store data, they can optimize query performance in some cases, especially if indexes are used efficiently on the underlying tables. Instead of running the entire query each time, a well-designed view can reduce the time it takes to retrieve results.

Applications of Views in Data Analysis

- **Presenting Summarized Data:** Views are commonly used to generate summary reports or aggregated data. You can create a view that pre-aggregates data, such as total sales, average scores, or counts, which can be used directly in reporting without re-writing the aggregation logic.

Example:

```
CREATE VIEW sales_summary AS
SELECT product_id, SUM(quantity_sold) AS total_sales
```

```
FROM sales
GROUP BY product_id;
```

- **Streamlining Reports:** In businesses, views are often created to serve as pre-formatted reports. A view can be set up with specific filtering conditions to generate reports for particular departments, regions, or other criteria without manually modifying queries.

Example:

```
CREATE VIEW region_sales AS
SELECT region, SUM(sales) AS total_sales
FROM sales
GROUP BY region;
```

- **Data Cleaning and Transformation:** When dealing with large datasets, cleaning and transforming data is essential. A view can be created to format and filter raw data, making it easier to work with in analysis. This is particularly useful when integrating data from multiple sources or standardizing formats across different datasets.

Example:

```
CREATE VIEW clean_data AS
SELECT TRIM(UPPER(name)) AS name, birth_date
FROM raw_data;
```

Real-World Use Cases

1. **Data Analysis Platforms:** In large organizations, data analysts use views to simplify access to key datasets, often creating views that aggregate data from different departments like sales, marketing, and HR.
2. **Reporting Systems:** Reporting tools often rely on views to generate daily, weekly, or monthly reports. By using views, businesses can ensure consistent and accurate reports are generated from the same underlying query.
3. **Access Control:** In financial institutions or healthcare systems, views are used to restrict access to sensitive data while allowing authorized users to access non-sensitive information.

Best Practices for Using Views

- **Keep it Simple:** Avoid overly complex views that could impact performance. Instead, use simple views to abstract commonly used queries.
- **Indexing:** Ensure that the underlying tables used in views are properly indexed to improve query performance.

- **Security:** Use views to enforce data security and access control by exposing only the necessary data to different users.

Conclusion

Views are a powerful tool in SQL that allow you to simplify complex queries, enhance data security, and optimize data access for analysis. By abstracting data and providing a clear, focused view, they play a critical role in effective data management and analysis workflows. Incorporating views into your data analysis process can greatly improve the clarity and efficiency of your operations.

Comments

7

Like Comment Share

Add a comment...

No comments, yet.

Be the first to comment.

Start the conversation

Enjoyed this article?

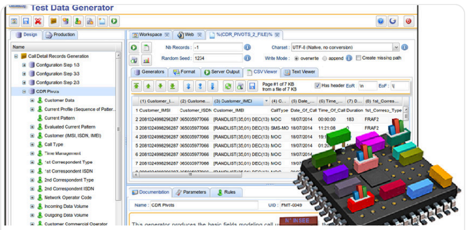
Follow to never miss an update.

Deepthy A

Data Analyst | Python • SQL • Power BI • Excel | Turning Data into Business Insights

Follow

More articles for you



Types of Data - 'DVT'

Key Features for Selecting or Building Effective Data Generation

Jean-Lin Pacherie, Ph.D.

7

What is a data domain — and why should you care?

Smart Data Governance

15 · 10 comments · 2 reposts

Sandeep Vupputuri

4