Nichole Jones
June 12, 2023
Foundations of Databases & SQl Programming

SQL Views

Introduction
In this paper I will be discussing SQL views, when you would use them and similarities and differences between a View, Function, and Stored Procedure.

When to use a SQL View
SQL views are virtual tables that are created based on the result of a query. They allow you to store and reuse complex queries as a named object in your database. Here are some scenarios where you would typically use SQL views:

1. Simplifying complex queries: If you have a query that involves multiple joins, aggregations, or calculations, creating a view can simplify the process of querying that data. You can encapsulate the complex logic within the view and then use the view as a simpler representation of the data.
2. Data security and access control: Views can be used to restrict access to sensitive data in a database. Instead of granting direct access to the underlying tables, you can create views that expose only the necessary columns or rows to different users or user groups. This allows you to implement fine-grained access control and provide a layer of security.
3. Data abstraction and encapsulation: Views can provide an abstraction layer over the underlying tables. They can hide the underlying table structure and provide a simplified and consistent interface for applications or end-users. This helps in decoupling the database schema from the application code, making it easier to modify the underlying schema without affecting the dependent applications.
4. Query reuse and modularity: By creating views, you can reuse complex queries in multiple places within your database or across different applications. This improves maintainability and reduces duplication of code. If the underlying query needs to be modified or optimized, you can update the view definition once, and all the queries that use the view will automatically reflect the changes.
5. Performance optimization: Views can be used to precompute and store the results of complex queries. This can improve performance by avoiding redundant computations. For example, if you have a query that involves aggregations or calculations, you can create a view that stores the precomputed results, and subsequent queries can directly retrieve data from the view instead of re-computing the results each time.
6. Data transformation and normalization: Views can be used to transform the structure of the data and normalize it according to specific requirements. You can combine columns from multiple tables, apply calculations, or aggregate data to create a view that

represents the desired transformed data. This allows you to present the data in a way that is more suitable for reporting or analysis purposes.

These are some of the common use cases for SQL views. They provide flexibility, improve query simplicity, enhance security, and enable better performance optimization in database management systems.

The differences between a View, Function, and Stored Procedure
There are several types of Joins used in SQL, here I will be discussing the similarities and differences between inner, outer, and cross joins.

An inner join returns only the rows that have matching values in both tables being joined. It combines rows from two tables based on the specified condition or common column between them.

An outer join returns all the rows from one table and the matching rows from the other table. It allows you to include unmatched rows from one table in the result set.

A cross join returns the Cartesian product of the two tables, which means it combines every row from the first table with every row from the second table.

Where the Inner Join returns only the matching rows, the outer join returns matching rows and potentially unmatched rows. An Outer join can be further categorized into left, right, and full join based on which table's rows are included in the result set. A cross join combines every row from one table with every row from the other table, resulting in a Cartesian product.

What is a Self-Join

Views, functions, and stored procedures are all database objects that serve different purposes. Here are the key differences between them:

1. Views:
    - Views are virtual tables that are created based on the result of a query. They don't store data themselves but provide a way to represent a subset of data from one or more tables.
    - Views can be used to simplify complex queries, provide data security by restricting access to certain columns or rows, abstract the underlying table structure, and enable query reuse and modularity.
    - Views are typically used for querying and presenting data in a specific way, but they cannot accept parameters or perform complex logic like calculations or data modifications.
2. Functions:

- Functions are database objects that accept input parameters, perform computations or operations on the input, and return a single value or a table of values.
- Functions can be used to encapsulate reusable computations, calculations, or transformations that can be applied to data. They are often used in queries, expressions, or other functions.
- Functions can be scalar functions that return a single value or table-valued functions that return a table of rows.
- Functions are generally read-only and cannot modify the data in the database directly.

3. Stored Procedures:
    - Stored procedures are a collection of SQL statements and procedural logic that are stored and executed on the database server.
    - Stored procedures can accept input parameters, perform complex business logic, execute multiple SQL statements, and return multiple result sets.
    - Stored procedures are often used to encapsulate business rules, implement data modifications, handle transactions, and perform batch operations.
    - Unlike views and functions, stored procedures can modify the data in the database, including inserting, updating, or deleting records.
    - Stored procedures can be called from applications or other stored procedures to execute a predefined set of actions.

Summary

In summary, views provide a way to represent a subset of data from tables, functions encapsulate reusable computations, and stored procedures are used to store and execute a collection of SQL statements and procedural logic, including data modifications