Name: VBetin

Date: Nov. 19, 2025

Course: IT FDN 130 A Au 25: Foundations Of Databases & SQL Programming

GitHub: https://github.com/valeria-betin/DB_SQL_Foundations

Assignment 06:

The Role of SQL Views, Functions, and Stored Procedures in Data Management

Introduction

Organizations and businesses worldwide have an increasing dependency on sophisticated data systems to manage their database. SQL Views, Functions, and Stored Procedures are three essential tools that developers and analysts use to structure, secure, and simplify access to relational data. Although they share similarities as entities that encapsulate SQL logic, each serves a distinct purpose within a database architecture. This paper explains when SQL Views should be used and examines the similarities and differences among Views, Functions, and Stored Procedures. As reporting demands grow it is important for organizations to maintain accurate and efficient databases that support internal operations. In a nonprofit business model, it's crucial to manage data points like donors, volunteers, communications, and financial contributions.

The Purpose and Use of SQL Views

A SQL View is a saved SELECT statement that can be queried as though it were a virtual table. Views do not physically store data, rather they provide a window into the underlying tables. This makes them valuable within a database where data integrity, access control, and reporting are crucial. One of the most common uses of a View is to support reporting functions. In a nonprofit that may be data that involves donors, their contributions, and allocation of funds. Often an analyst will need to combine multiple tables to prepare summaries for the directors of an organization to guide their fundraising strategies. Without Views, an analyst would have to repeatedly write complex JOIN statements every time a report is needed. By capturing those queries in a View, an organization can ensure consistency and reduce the likelihood of human error because the

Views being used have been tested and tried so in theory they should always work, barring any unforeseen circumstances. For instance, a View that joins donors to their donations might be created as follows:

```
CREATE VIEW vDonorDonations as
SELECT d.DonorID,
       d.FirstName,
       d.LastName,
       dn.Amount,
       dn.DonationDate
FROM Donors as d
      INNER JOIN Donations as dn
      ON d.DonorID = dn.DonorID;
```

Once created, this View can be used by staff with minimal SQL knowledge to obtain reliable donor contribution data without exposing the underlying table structures.

Another significant use of Views is to provide an abstraction layer that protects the physical design of the database architecture itself. In well-governed databases, direct access to tables is limited to a select few individuals known as administrators. End users and applications that use the database, instead query base-level Views that expose only the fields they need to see. This type of approach to database management enhances security by hiding sensitive columns—such as payment information—and helps maintain consistency when table structures change. For example, if the Donors table required a redesign to support additional metadata, only the View would need modification, allowing dependent reports and applications to run without interruption.

Views can also be schema-bound to prevent accidental changes to the underlying tables. When a View is created with schema binding, SQL Server ensures that structural changes to the referenced tables do not invalidate the View. This is especially useful for organizations that depend on long-term reporting stability. For example, for a nonprofit that could be reporting that's for compliance and multi-year grant evaluations. A schema-bound version of the donor–donation View might look like this:

```
CREATE VIEW vDonorDonations
WITH SCHEMABINDING
as
SELECT d.DonorID,
       d.FirstName,
       d.LastName,
       dn.Amount,
       dn.DonationDate
FROM dbo.Donors as d
       INNER JOIN dbo.Donations as dn
       ON d.DonorID = dn.DonorID;
```

As a result of adding the schema binding in the query above, no one would be able to drop or alter the Donors or Donations table in a way that breaks this View, thereby protecting critical reporting systems.

Comparing Views, Functions, and Stored Procedures

While SQL Views play a critical part in abstraction and reporting, Functions and Stored Procedures extend the capability of the database by enabling constraints, logical operations, and data manipulation. Although they all contain SQL code, their purpose differs significantly.

Views and Functions

Views and Functions are often compared because they can both return data that resembles a table. Functions, however, introduce features that Views do not have. A critical distinction here is that Functions can accept parameters, thereby allowing users to generate context specific results. In the nonprofit example, this can be extremely useful to where donor specific or filtered summaries may be required. While Views must be filtered externally using a WHERE clause, a Function can produce a parameterized dataset immediately upon being queried.

Functions can also return scalar values – for example there may be a need to determine a donor’s total contributions for the year for a tax receipt, this would require the use of a scalar function to pull in the SUM of the amount rather than the individual donations. Once created, this function can be embedded directly into SELECT queries, which Views and Stored Procedures cannot support. These capabilities allow Functions to be more versatile for analytical tasks, though they are still constrained in comparison to Stored Procedures because Functions are not able to modify data or execute multiple SQL statements.

Views and Stored Procedures

Views and Stored Procedures share the ability to encapsulate SELECT statements, but Stored Procedure have more power in that a Stored Procedure can include conditional logic, variables, transaction management, and multiple SQL statements. This essentially allows a developer to have the ability to encode a full business process into a single callable object or stored procedure. In a nonprofit something like this might look like the automation of complex workflows such as recording donations, sending receipts, or updating donor recognition tiers.

A Stored Procedure might insert a donation and then evaluate whether the donor qualifies for a new giving category and should be moved into say the Major or Mid Portfolio. Views cannot perform any of these actions because they are limited to read-only, table-like expressions. Stored Procedures cannot be embedded within SELECT statements, whereas Views function as queryable objects. Thus, Stored Procedures are operational tools, while Views are representational tools.

<u>Functions and Stored Procedures</u>

Functions and Stored Procedures share some similarities in that they both support parameters and can incorporate logical constructs, for example an IF statement. However, the purpose of the two is drastically different. Functions always return either a single value or a table, and they must always return the same results given the same inputs, meaning they cannot modify the state of the database itself. Stored Procedures on the other hand do not have these limitations; they may return multiple result sets, modify tables, etc. Conclusively, this makes Functions a preferable tool for reusable calculations, while Stored Procedures are usually reserved for operational tasks like data imports, bulk updates, or nightly maintenance routines like account deduplication.

Conclusion

SQL Views, Functions, and Stored Procedures each contribute uniquely to an organizations data infrastructure. Views allow for a stable abstraction layer that can simplify reporting, enhance security, and shield applications from schema changes. Functions expand the analytical capability of a database through parameterized expressions and reusable logic. Stored Procedures facilitate operational workflows by supporting data modification and procedural logic. Together these tools allow for a nonprofit organization to manage donor relationships, track financial contributions and grant specific designations. Understanding how and when to use each of these objects is key to maintaining a reliable, efficient, and secure data environment capable of supporting an organizations mission.