Luis Valderrama

May 13, 2021

IT FDN 130 A

Module 06

SQL Views

Intro

During the course of the Sixth module, I learned the basic concepts and practical uses of Views, and an instruction to Functions and Stored Procedures available in the MS SQL RDMS to enable the user(s) to store queries in a database. This document addresses uses, similarities and differences between Views, Functions and Stored Procedures in addition to examples of Views used in the database created for this assignment, *Assignment06DB_LuisValderrama,* contrasting with Functions and Stored Procedures.

SQL Views versus Functions, and Stored Procedures

Views

"A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view." *https://www.tutorialspoint.com/sql/sql-using-views.htm, (2017) (external).* **A data analyst would primarily use BASE or BASIC VIEWS to:**

- Store codes in a database.
- Narrow the scope, simplify, and modify the visualization of the database. And, summarize the data from various tables used for generating reports.
- Serve as security measure, allowing users to access the data via the View without having to grant permissions to access tables directly. And, restrict access to the data in such a way that a user can see and, in cases modify exactly what they need and no more.
- Structure data in a way that users or classes of users find natural or intuitive.
- Copy data in SQL Server to improve performance and to partition data.

At a high level, the types of Views are:

- **Indexed Views.** The View definition has been computed and the resulting data stored just like a table. This process is also known as the View has been "Materialized".
- **Partitioned Views.** Joins horizontally partitioned data from a set of member tables across one or more servers, making the data appear as if from one table.
- **System Views.** Return information about the instance of SQL Server or the objects defined in the instance.

The example provided below on (figure 1.1) used in *Assignment06DB_LuisValderrama*, (2021) (SQL script) displays the structure of a Base View using CREATE VIEW, AS statement, followed by the name of the view which in this case is 'v' plus the table name 'vProducts', making it comprehensive to the users. Additionally, the command WITH SCHEMABINDING protects the tables from changing in a way that would interfere with the performance of the View. The schema binding requires the use of a two-part name of the table, in this case dbo.Products. Views are used with SELECT FROM statement followed by the table's attributes. The View batch is contained within the GO command.

```sql
--Establishing a View for Products table

GO
CREATE VIEW vProducts
WITH SCHEMABINDING
AS
  SELECT ProductID, ProductName, CategoryID, UnitPrice
  FROM dbo.Products;
GO
```

FIGURE 1.1: Creating a Base View with schema binding option.

Views can follow permission protocols by using DENY or GRANT commands. The example presented below on (figure 1.2) displays the option of completely restricting public access to the actual Products table while granting public access to the View Products 'vProducts' table. This operation is critical to protect confidential information.

```sql
--Permission setup for Products Table

DENY SELECT ON Products TO PUBLIC;
GRANT SELECT ON vProducts TO PUBLIC;
GO
```

FIGURE 1.2: Example of allowing public access to the Products View while restricting access to the Products table itself.

Note: Views are generally used in simpler settings. However, Views may be used in some complex settings as well. In the example below on (figure 1.3) a View is created to return information from two tables, Categories and Products by way of JOIN, combined by ORDER BY clause to specify the order of attributes to which the data will be displayed, in this case CategoryName and ProductName. This requires the use of TOP + value clause after the SELECT clause as Views cannot dictate how the data will is sorted.

```sql
--Establishing a View to show a list of Category, Product names and priced

GO
CREATE VIEW vProductsByCategories
AS
  SELECT TOP 1000000
    CategoryName
  , ProductName
  , UnitPrice
  FROM Categories
  INNER JOIN Products
  ON Categories.CategoryID = Products.CategoryID
ORDER BY CategoryName, ProductName ASC;     --Ordered by category name, then by product name
GO
```

FIGURE 1.3: Example of View using a JOIN and in combination with ORDER BY clause.

In general. Views, Functions and Stored Procedures are very similar in essence. However, some differences exist. Below, presented in (table 1.1) are some of the primary similarities and differences.

Primary Similarities and Difference of Views, Functions, Stored Procedures	
Similarities	**Differences**
• **Store and Reuse of Codes.** Views, Functions and Stored Procedures eliminate the need to rewrite the same codes, decreases code inconsistencies and allows the code to be accessed and executed by any user.	• Unlike Views, a Function will return an individual value.
• **Enhanced Efficiencies:** ○ Reduce server/client networks traffic. ○ Enable faster execution. ○ Improve performance. ○ Efficient maintenance.	• Unlike Views, with Functions you must use 'RETURN' after the 'AS' keyword, and "()" after the name of the Function
• **Stronger Security.** Various users can perform operations on underlying database objects through a View, Function and Procedure even in the event that users have no permissions on those underlying objects.	• Unlike Views, a Function can be used as an expression.
• **Ease of Data Visualization.** Views, Functions and Stored procedures narrow the scope, and simplify the visualization of the database. And, summarize the data from various tables used for generating reports.	• Views are used with SELECT statement only. While, Stored Procedures hold more complex logic such as INSERT, DELETE and UPDATE.
• **Similar Creation.** Views, Functions and Stored Procedures are established similarly using the CREATE statement.	• Unlike Views, a Stored Procedure requires an EXECUTE statement.

TABLE 1.1: Similarities and differences between Views, Functions and Stored Procedures.

In contrast to Views, the example presented below on (figure 1.4) used in the Mod06Lab displays the structure of a Function created to return information from two joined tables, Customers and Orders along with various functions. Unlike Views, **Functions are used in more complex situations**, while similar to Views, Functions are contained within GO commands, are established using CREATE statements, and store codes. Functions must include RETURNS clause to define the data to be displayed.

```
GO
CREATE FUNCTION dbo.fNumberOfCustomerOrdersByLocationAndYears()
RETURNS TABLE
AS
  RETURN(SELECT CustomerName = CompanyName
      , City
      , Region = ISNULL(Region, Country)
      , Country
      , NumberOfOrders = COUNT(OrderID)
      , OrderYear = YEAR(OrderDate)
FROM Northwind.dbo.Customers
JOIN Northwind.dbo.Orders
ON Northwind.dbo.Customers.CustomerID = Northwind.dbo.Orders.CustomerID
GROUP BY CompanyName
      , City
      , ISNULL(Region, Country)
      , Country
      ,YEAR (OrderDate));
GO
```

FIGURE 1.4: Example of a Function.

Another contrast to Views, the example presented below on (figure 1.5) used in the Mod06Lab. Generally, **Stored Procedures are used in more complex situations**. The structure of a Stored Procedure established to execute a process. Similar to Views and Functions, Stored Procedures are contained within GO commands and are established by using CREATE statements. However, Stored Procedures are also contained within BEGIN and END transaction statements, defining the start and conclusion of the procedure. To execute the procedure the user must use the key word EXECUTE or (EXEC) followed by the Stored Procedure.

```sql
GO
CREATE PROCEDURE dbo.pCustomersByLocation
AS
 BEGIN
  SELECT CustomerName = CompanyName
  , City
  , Region = ISNULL(Region, Country)
  , Country
  FROM Northwind.dbo.Customers;
 END
GO

EXEC pCustomersByLocation;
GO
```

FIGURE 1.5: Example of a Stored Procedure

Summary

To Recap, the sixth module taught me the basic concepts and practical uses of Views, and offered an introduction to Functions and Stored Procedures available in the MS SQL RDMS to enable the user(s) to store queries in a database. Functions and Stored Procedures will be examined in greater detail in the upcoming modules. The database created for this assignment, Assignment06DB_LuisValderrama, primarily applies functionalities learned throughout prior modules and the practical application of SQL Views.