**Name:** mGrilli

**Date:** 11/20/2022

**Course Name:** IT FDN 130

**GitHub URL:**

**Assignment 06 – VIEWS**

**Summary**

This week we once again expanded our SQL knowledge base by incorporating VIEWS. We learned what a view was, when and why to use them, and their benefits. We also touched on two other methods, User Defined Function (UDF), and Stored Procedures. Though at face value they function similarly, we learned that there are some major differences. By using VIEWS, UDF, and Stored Procedures, we can make our code more secure and easier to use.

**Topic 1**: When to use SQL Views

An SQL VIEW is a virtual table which contains the values of a query. Basically, a VIEW is a saved SELECT statement that lives in the database alongside the tables. In addition to all the system VIEWS, there should be base views. A base view should display all the columns and rows from the table the VIEW is being used.

CREATE VIEW [View Name]

WITH SCHEMABINDING [Keeps tables from changing]

AS SELECT [Table Columns]

FROM [Table]

***Figure 1: SQL VIEW structure (Base View)***

When working with a data base that has lots of data and lots of users, its important to set up permissions to protect the integrity of that data. When you assign a user permission to a VIEW, that user will only be able to reference the virtual table and not the actual table.

--Deny Access to Group

DENY SELECT ON [Table] TO [Group]

--Grant Access to group

GRANT SELECT ON [Table] TO [Group]

***Figure 2: SQL Denying and granting permissions***

In the real world a SQL developer will create Reporting Views that can generate elaborate reports for the user without said user having to see a single line of the query.

CREATE VIEW vInventoriesForChaiAndChangByEmployees

WITH SCHEMABINDING

AS

SELECT

    Categories.CategoryName,

    Products.ProductName,

    Inventories.InventoryDate,

    Inventories.Count,

    Employees.EmployeeFirstName + ' ' + Employees.EmployeeLastName

AS

EmployeeName

FROM

    dbo.Categories

JOIN

    dbo.Products

ON

    Categories.CategoryID = Products.CategoryID

JOIN

    dbo.Inventories

ON

    dbo.Products.ProductID = Inventories.ProductID

JOIN

    dbo.Employees

ON

    Inventories.EmployeeID = Employees.EmployeeID

WHERE

    ProductName = (

        SELECT

            ProductName

        WHERE

            Products.ProductName = 'Chai'

OR

Products.ProductName = 'Chang')

GO

***Figure 3: Example of creating a Reporting View***

That user would write the following SELECT statement to which references the query statement that lives on database and in the background.

SELECT \* FROM [dbo].[vInventoriesForChaiAndChangByEmployees]

ORDER BY 3,1,2

***Figure 4: Example of how the user would access the granted view***

**Table

Description automatically generated**

***Figure 5: Result of running the SELECT \* FROM [dbo].[*vInventoriesForChaiAndChangByEmployees].**

**Topic 2:** The differences andsimilarities between a View, User Defined Function (UDF), and Stored Procedures.

VIEW – An SQL VIEW is a virtual table which contains the values of a query. Basically, a VIEW is a saved SELECT statement that lives in the database alongside the tables. Views are like inline table functions in that they are better for performance, and you can use the results in another query. Another benefit of a view is that you can index it and cache that data upfront which makes them preform even faster. Views also have the benefits of granting and denying permissions, but unlike Stored Procedures and Functions, cannot take parameters which makes them a little less flexible.

UDF – The SQL Server user-defined functions help you simplify your development by encapsulating complex business logic and make them available for reuse in every query. One such functions is the Scaler Function which will only return one value. Because it does these operations for each query item, running this will devour a lot of resources. Then there’s the in-line table function which allows you to reuse the logic, like a subquery, over add over again without having to rewrite any code. Lastly there’s the Multi Statement Value Functions which allows you to run many statements at once.

Stored Procedure – The SQL Stored Procedure is a group of statements that can be run repeatedly. Based on the parameters you set, you can preform one or many Data Manipulation Language (DLM) and return a value if there is one to return. Stored Procedures are like the Multi Statement Value Function in that you can execute many statements at once but cannot use the result returned in another query. Like views they are more secure because you can grant and deny permissions to users.

The main similarities between the three is that these are generated and then called upon later by invoking it. They all contain predefined code that the front-end user can access via the name and declaration for the respective method being used.

To get a better idea of how they differ, lets take a second to look at each of their individual structure.

*--VIEW*

CREATE VIEW

vCategories

    WITH

SCHEMABINDING

    AS

    SELECT

        CategoryID,

        CategoryName

    FROM

        dbo.Categories;

    GO

*--UDF*

CREATE FUNCTION

dbo.MultiplyValues(@Value1 Float, @Value2 Float);

RETURNS

Float

  AS

  BEGIN

  RETURN(

SELECT

@Value1 \* @Value2

);

  END

GO

*-- CALL THE FUNCTION*

SELECT

Tempdb.dbo.MultiplyValues(4, 5);

GO

*--STORED PROCEEDURE*

CREATE PROCEDURE

pProducts()

AS

    SELECT

        ProductID,

        ProductName,

        CategoryId,

        Discontinued

      FROM

        Northwind.dbo.Products;

GO

EXECUTE

    pProducts();

GO

***Figure 6: structures for VIEW, UDF, and Stored Procedure.***

**Conclusion**

As an SQL developer we have many tools at our disposal to give users the ability to run complex queries with only a few lines of code. These include VIEWS, UDF, and Stored Procedures. Although they have similarities, they each come with their limitations and benefits. Some can do complex operations but eat up resources and others use less resources but aren’t as flexible or dynamic. When determining which method to use, it is very important to understand the problem at hand, so that you can determine the best way forward.