Name: Marcus Bradlee

Date: November 19, 2025

Course: IT FDN 130

GitHub: DBFoundations

Assignment 6 – Views

Introduction

Assignment 6 introduced us how to further our layers of abstraction between the data in our database and our client-side end users. To maintain layers of abstraction, we can use tools such as Views, Functions, and Stored Procedures. We can also use tools such as Schema Binding and permission manipulation to ensure the integrity of our Base data.

Views

Views may seem like tables on the surface, but they are actually saved select statements, typically on the more complex side. Views do not provide an increase in performance as they do not have separate data pages, and they are literally stored as compiled text.

Views are mainly created for convenience, but they also provide a layer of abstraction making it harder for the original data in the tables from being tampered with. It is generally recommended that you do not program directly into a table, so views can be a good way to gather, join, and analyze the data in your tables.

Creating a View

After you have created tables with data within your database, you can create a select statement and declare it as a new view, effectively saving your select statement for later use. Here is a simple example from this week's assignment:

```
Create View vProductsByCategories
As Select Top 1000
      c.CategoryName
      ,p.ProductName
      ,p.UnitPrice
From Categories c
Join Products p On c.CategoryID = p.CategoryID
Order By c.CategoryName, p.ProductName, p.UnitPrice
```

In this example we are writing a query with our categories table and our products table to display products, their categories, and their prices, and saving it to a view called vProductsByCategories. For further abstraction, you can turn Categories and Products into views and use those for the Select query instead. This is called a Base or Basic view and is the standard practice for abstraction.

Another notable aspect of the above example is the use of the Order By clause. By design, views cannot dictate how the selected data is stored. However, in some RDMS you can use the Top clause to trick the system into allowing an order by statement. That being said, typically best practice would be to use the order by clause in your final select statement after creating the view.

Schema Binding

Foreign keys will not protect a parent table from being dropped, so it is important to create views using an option called Schema Binding. Schema Binding keeps tables from changing so much that the view does not work anymore. You would typically begin by making your Base Views with Schema Binding and then protect the private data with permissions, ensuring that no changes can be made that would corrupt your view. For example:

```
Create View vCategories With SchemaBinding
As Select
       CategoryID
       ,CategoryName
From dbo.Categories;
Go

Deny Select On Categories To Public;
```

In the above example, we begin by creating a Base View for Categories called vCategories with Schema Binding, meaning any changes to Categories that would break vCategories will not be allowed. Then we use Deny Select to protect Categories data with permissions and ensure the only way to access this data for non admins to access this data is through Views.

Functions

In addition to SQL's built in functions, you are also able to create your own custom functions, also known as User Defined Functions (UDFs). There are two basic kinds of functions:

- Functions that return a table of values
- Functions that return a single value

Functions are similar to Views in that functions are able to return a table of values using a select statement. For example:

```
Create Function dbo.fProducts()
Returns Table AS
Return(
       Select ProductID, ProductName, CategoryId
       From dbo.Products
);
```

```
Select * from dbo.fProducts();
```

The function example above would return a full dataset as a table containing the ProductID, ProductName, and the CategoryID from Products, similar to how you could create a View of Products using a similar select statement.

The difference between Functions and Views lies in Functions ability to accept parameters to change the results of the query. For example:

```
Alter Function dbo.fProducts(@CategoryId int)
Returns Table AS
Return(
	Select ProductID, ProductName
	From Northwind.dbo.Products
	Where CategoryID = @CategoryId
);

Select * From dbo.fProducts(1);
```

The example above demonstrates the ability to use a functions parameter to pass in a value to determine which data is being output from the select query. In this case, we are using the parameter to specify that the resulting data set should only contain products with a CategoryID of 1.

Stored Procedures

Similar to Views and Functions, Stored Procedures (Sprocs) are a named set of SQL statements. Creating a Sproc is similar to creating a View or Function:

```
Create Procedure pProducts() As
Select
	ProductID
	,ProductName
	,CategoryId
From dbo.Products;
Go

Execute pProducts();
```

Also similar to Functions, Sprocs can be passed one or multiple parameters which will affect the results of the saved query. The main advantage of Sprocs is that they allow you to do anything you need to within SQL, for example inserts, updates, deletes, and selects.

Summary

Abstraction layer objects such as views make changes to a table design easier. With Views, Functions, and Stored Procedures, we are able to continue to add layers of abstraction to maintain a clean and secure database. Views help us maintain applications access data, Functions help use reuse our code to improve database efficiency, and Stored Procedures help us automate our processes.