Name: Katarzyna (Kasia) Swica
Date: November 23, 2022
Course: IT FDN 130 A Au 22: Foundations of Databases and SQL Programming
Github Link: https://github.com/kswica-uw/DBFoundations

Assignment 6- SQL Views

Introduction

This document details different scenarios in which a user would use a SQL View as well as the similarities and differences between SQL Views, Functions, and Stored Procedures.

Applications and Uses of SQL Views

SQL Views can be thought of as named, saved SELECT statements. When a SQL View is run, data is retrieved at that time, ensuring that the data from the View is always current. However, the underlying data, the base relations of the table, are generally not accessible via a View.

Views are helpful tools for access management, particularly when they are created with constraints (i.e. not all the data is retrieved) and clear access management (i.e. users have limited access to information in the database). Providing users of a database with access to Views is more secure than enabling full access to the database tables because Views are generally a subset of the full table(s), and the ability for users to make changes to the data is limited. For example, while a company may have a table of all employee data, a public view of this data could be created to exclude sensitive information, such as SSN or salary.

Views are also helpful tools for abstracting and simplifying queries, particularly when the query is used often and by a variety of users. A complex query can be saved as a View, obscuring the underlying code. In this way, novice users don't have to recreate or understand the underlying code to retrieve results. Even for advanced users, queries that reference Views can result in code that is faster to write and simpler to read (https://www.youtube.com/watch?v=Y-Qk4vpklJ8&list=PLfycUyp06LG8cefs0gA38wO7nFrRjD5Ad&index=7, 2022).

For example, a complex query such as this, which spans multiple tables and includes constraints, can be consolidated into a view:

```sql
CREATE VIEW vInventoriesByProductsByCategoriesByEmployees
    AS
        SELECT TOP 100000000
            c.CategoryID,
            c.CategoryName,
            p.ProductID,
            p.ProductName,
            p.UnitPrice,
            i.InventoryID,
            i.InventoryDate,
            i.[Count],
            e.EmployeeID,
            e.EmployeeFirstName+' '+e.EmployeeLastName AS [Employee],
            m.EmployeeFirstName+' '+m.EmployeeLastName AS [Manager]
            FROM vCategories AS c
                JOIN vProducts AS p
                    ON c.CategoryID=p.CategoryID
                JOIN vInventories AS i
                    ON i.ProductID=p.ProductID
                JOIN vEmployees AS e
                    ON e.EmployeeID=i.EmployeeID
                JOIN vEmployees as m
                    ON m.EmployeeID=e.ManagerID
            ORDER BY c.CategoryName, p.ProductName, i.InventoryID, [Employee];
GO
```

Figure 1 Example of a complex SQL SELECT statement simplified into a View.

```sql
Select * From [dbo].[vInventoriesByProductsByCategoriesByEmployees]
```

Results Messages

	CategoryID	CategoryName	ProductID	ProductName	UnitPrice	InventoryID	InventoryDate	Count	EmployeeID	Employee	Manager
1	1	Beverages	1	Chai	18.00	1	2017-01-01	39	5	Steven Buchanan	Andrew Fuller
2	1	Beverages	1	Chai	18.00	78	2017-02-01	49	7	Robert King	Steven Buchanan
3	1	Beverages	1	Chai	18.00	155	2017-03-01	59	9	Anne Dodsworth	Steven Buchanan
4	1	Beverages	2	Chang	19.00	2	2017-01-01	17	5	Steven Buchanan	Andrew Fuller
5	1	Beverages	2	Chang	19.00	79	2017-02-01	27	7	Robert King	Steven Buchanan
6	1	Beverages	2	Chang	19.00	156	2017-03-01	37	9	Anne Dodsworth	Steven Buchanan
7	1	Beverages	39	Chartreuse verte	18.00	39	2017-01-01	69	5	Steven Buchanan	Andrew Fuller
8	1	Beverages	39	Chartreuse verte	18.00	116	2017-02-01	79	7	Robert King	Steven Buchanan
9	1	Beverages	39	Chartreuse verte	18.00	193	2017-03-01	89	9	Anne Dodsworth	Steven Buchanan
10	1	Beverages	38	Côte de Blaye	263.50	38	2017-01-01	17	5	Steven Buchanan	Andrew Fuller
11	1	Beverages	38	Côte de Blaye	263.50	115	2017-02-01	27	7	Robert King	Steven Buchanan
12	1	Beverages	38	Côte de Blaye	263.50	192	2017-03-01	37	9	Anne Dodsworth	Steven Buchanan

Figure 2 Sample results of the View created in Figure 1.

As well, since the View is an abstraction of underlying base tables, the name and format of columns can be changed. This can be a helpful way to clarify and tailor the readability of the data for different users.

Finally, Views are helpful ways to provide information to applications in a consistent way even when changes occur in the database. A WITH SCHEMEBINDING clause can be added to a View to ensure that a linkage between an underlying base table and the view is not broken, for example by data deletion (RRoot, Module06Notes, 2022). Views can maintain the way applications access data, even when the database tables need to be changed (RRoot, Module06Notes,2022).

Comparing Views, Functions, and Stored Procedures

Views, Functions, and Stored Procedures can all be thought of as named SQL statements.

The syntax for a Function is unique from a View. The alias of the database (i.e. 'dbo' prefix below) must be included when creating or referencing a Function, for example.

```
-- View
Create View vCategories
  AS
    Select CategoryID, CategoryName, [Description], Picture,
     From Northwind.dbo.Categories;
go
Select * from vCategories;
Go

-- Function
Create Function dbo.fCategories()
 Returns Table
 AS
   Return(
    Select CategoryID, CategoryName, [Description], Picture
   From Northwind.dbo.PCategories);
go
Select * from dbo.fCategories();
Go
```

Figure 3 Syntax of a View compared to the syntax of a Function.

User defined functions (UDFs) come in two types: (a) functions that return a table of values and (b) functions that return a single value (RRoot, Module06Notes, 2022). Table functions are very similar to Views.

Functions, unlike Views, can include parameters that change the results of a query. This enables the user to place an argument such as a multiplication statement, into the parameters of the function. Scalar functions can also be used within SELECT and/or WHERE clauses. In this way, the user can have the Function return the single value result of an expression where the inputs are from different columns or tables. These types of custom scalar functions can be helpful as check constraints in a database.

```
472  USE Pubs;
473
474  Go
475  Create Function dbo.HighRoyalty(@Value1 Float, @Value2 Float)
476   Returns Float
477   As
478    Begin
479     Return(Select @Value1 * @Value2);
480    End
481  go
482
483  Select
484   ,dbo.HighRoyalty (hirange,royalty) as HighRoyalty
485  From dbo.roysched;
486  go
```

Figure 4 Sample scalar function included in a SELECT statement.

Store Procedures

Unlike Views and Functions, Stored Procedures can include not only SELECT statements, but also other statements such as INSERT, UPDATE, and DELETE. Stored Procedures can even include many different SQL statements.

Like Functions, Stored Procedures can have parameters. However, Stored Procedures themselves cannot evaluate into anything (i.e. you can't evaluate a multiplication statement in a Stored Procedure). Notably, Stored Procedures execute, unlike Views and Functions that "return" or "select from" (https://www.youtube.com/watch?v=22yz763fAg0&list=PLfycUyp06LG8cefs0gA38wO7nFrRjD5Ad&index=5, 2022).

```
-- Stored Procedure
Create Procedure pCategories()
AS
    Select CategoryID, CategoryName, [Description], Picture
    From Northwind.dbo.Categories;
go
Execute pCategories();
Go
```

Figure 5 Sample syntax for a SQL Stored Procedure.

There are system provided Stored Procedures such as sp_helptext that come by default with database management software. One should be careful not to override these defaults. Overall, the flexibility of Stored Procedures make them good tools for complex reporting.

Conclusion

Views, Functions, and Stored Procedures are tools that save SQL statements in a database. These abstraction tools have a variety of uses including increasing database security, streamlining reporting, and enabling more complex querying and reporting.