Name: Rhyss Garvida

Date: 5/30/2022

Course: IT FDN 130 A

Assignment 07

GitHub: https://github.com/rhyssg/DBFoundations-Module07

User-Defined Functions

Introduction

In the Server Query Language (SQL), a User-Defined Function is an object that allows for scripts to be "stored in the database [to] avoid writing the same code over and over again" (SQL Shack, https://www.sqlshack.com/learn-sql-user-defined-functions/, 2020) (External Site). It enables a user to control the input parameters and how it is defined to reach the desired output. The three types of User-Defined Functions are Scalar, Inline, and Multi-Statement Functions which can contain an increasing complexity of statements to yield a singular or unique set of values. The syntax for the object depends on whether the Create, Alter or Drop is used and can encapsulate SQL statement instructions that are either small or bulky (Figure 1). Thus, it is beneficial in situations where the function can be recalled for "calculation you'll repeat throughout your database" (SQL Shack, (SQL Shack, https://www.sqlshack.com/learn-sql-user-defined-functions/, 2020) (External Site).

```
CREATE FUNCTION [database_name.]function_name (parameters)
RETURNS data_type AS
BEGIN
    SQL statements
    RETURN value
END;

ALTER FUNCTION [database_name.]function_name (parameters)
RETURNS data_type AS
BEGIN
    SQL statements
    RETURN value
END;

DROP FUNCTION [database_name.]function_name;
```

Figure 1: Create/Alter/Drop Syntax for User-Defined Function

Scalar, Inline and Multi-Statement Functions

A User-Defined Scalar Function takes any number of parameters and "returns a single value each time it is invoked" (IBM, https://www.ibm.com/docs/en/db2-for-zos/11?topic=function-sql-Scalar-functions, 2022) (External Site). Much like the other User-Defined Functions, it is used to simplify code when complex and bulky calculations appear in many queries. The syntax encapsulates SQL statements designed to output a singular value by the defined function (Figure 2). As an example, the UCASE Scalar

Function returns each column value per row in uppercase (Figure 3). As such, it operates on each record independently and is based on user input to return a single value.

```
CREATE FUNCTION [schema_name.]function_name (parameter_list)
RETURNS data_type AS
BEGIN
    statements
    RETURN value
END
```

Figure 2: Scalar Function Syntax

```
SELECT UCASE(column_name) FROM table_name;
```

Figure 3: UCASE() Scalar Function

A User-Defined Inline Function "is a table expression that can accept parameters, perform an action and provide as its return value, a table" (SQL Server Central, https://www.sqlservercentral.com/articles/creating-and-using-Inline-table-valued-functions, 2020) (External Site). It is a table-valued function that encapsulates the code in a single executable database object allowing the user-defined object to be reused many times. It is much like a View but with the ability to accept parameters. The syntax involves a user-defined variable followed by the parameter and data type. It is then returned as a table displaying the output as defined by the return statements (Figure 4). As an example, the 'Sales.fn_OderDetails_IF" User-Defined Inline Function takes on the @SalesOrderID parameter values and returns it as a table based on the select statement definition (Figure 5). Thus, it is a great way to detail current table data and present them in varying ways for informational view.

```
CREATE FUNCTION [dbo].[udfGetProductList]
(@SafetyStockLevel SMALLINT
)
RETURNS TABLE
AS
RETURN
(SELECT Product.ProductID,
        Product.Name,
        Product.ProductNumber
 FROM Production.Product
 WHERE SafetyStockLevel >= @SafetyStockLevel)
```

Figure 4: Inline Function Syntax

```
CREATE FUNCTION Sales.fn_OrderDetails_IF(@SalesOrderID int)
RETURNS TABLE
AS

RETURN(
    SELECT ROW_NUMBER() OVER (
            PARTITION BY detail.SalesOrderID
            ORDER BY SalesOrderDetailID) AS OrderRow,
        detail.OrderQty, detail.UnitPrice, detail.LineTotal,
        prod.[Name] AS ProductName, prod.Color,
        prod.StandardCost, prod.[Weight]
    FROM Sales.SalesOrderDetail AS detail
    INNER JOIN Production.Product AS prod ON
        detail.ProductID=prod.ProductID
    WHERE detail.SalesOrderID=@SalesOrderID
);

GO
```

Figure 5: Inline Function Example

Lastly, a User-Defined Multi-Statement Function much like an Inline Function would return a table type result set, but with the difference in that the table is explicitly constructed in the script. The syntax involves defining the table structure to be returned followed by the function body encapsulated by a Begin and End (Figure 6). Here, the Name and DateOrdered columns are created and will be returned by the getCustomerAndDate() function, where the columns are defined by the encapsulated select statement within the Begin-End Statements (Figure 6). Therefore, this allows the user to process unique and focused business logic by constructing a virtual table on the fly.

```
CREATE FUNCTION getCustomerAndDate()
RETURNS @CustAndDateInfo TABLE
(
Name VARCHAR(10),
DateOrdered DATETIME
)
AS
BEGIN
INSERT INTO @CustAndDateInfo
SELECT C.FirstName, O.OrderDate
FROM Orders AS O
INNER JOIN Customers AS C
on O.CustID = C.CustID
RETURN
END
```

Figure 6: Multi-Statement Function Syntax and Sample

Summary

The goal is not to repeat yourself, which would otherwise make your SQL code bulky and hard to read. User-Defined Functions give the ability to encapsulate these SQL scripts, allowing for recall whenever desired informational value needs to be transformed by these objects. Depending if a single or tabular value is desired, Scalar, Inline, and Multi-Statement Functions are tools that can reduce repetitions and increase performance of the SQL code generated.