Name: NSchafi
Date: 8/23/2022
Course: IT FDN 130 A Su 22: Foundations of Databases & SQL
GitHub URL: https://github.com/njschafi/DBFoundations-Module07

Assignment 7 – Functions

Introduction

In this document, we will be going over the use and concepts of functions within SQL. SQL has numerous pre-built functions that are at the disposal of the user but there are times when there is no available function to perform a task. This is where User-Defined Functions (UDF) enter the room, they allow users to create their own functions. Functions provide a way for users to not have to re-write code and allow for ways to manipulate data.

SQL User-Defined Function (UDF)

As discussed briefly in the introduction, a User-Defined Function is basically a custom function created by a user. There are two basic types of functions; functions that return a table of values and functions that return a single value. A UDF is useful when a user wants to manipulate data in a specific way but SQL does not have an already pre-defined function to do that. Example 1 below, shows the syntax of a UDF:

```
Create Function dbo.MultiplyValues(@Value1 Float, @Value2 Float)
 Returns Float
 As
  Begin
   Return(Select @Value1 * @Value2);
  End
go
-- Calling the function
Select Tempdb.dbo.MultiplyValues(4, 5);
go
```

Example 1 - Syntax of a Scalar UDF

In Example 1, a function is created to multiply 2 values. You can see at the bottom, that the function (MultiplyValues) is called using (4, 5). The result, or the return of the function, will be a table showing a single value of 20.

Scalar, Inline and Multi-Statement Functions

A **Scalar** function in SQL, is a function that takes one or more parameters and returns a single value. Example 1 in the prior section, showcases this by multiply 2 numbers/parameters and outputting a single value.

Unlike a Scalar function, an **Inline** function returns a result-set rather than a single value. A user can pass in parameters into an Inline function and get a result table in return. The syntax for writing an inline function is shown in Figure 1:

```
CREATE FUNCTION fnNameOfFunction(

  -- parameters go here
  @param1 datatype,
  @param2 datatype, ...

)
RETURNS TABLE
AS
RETURN

  -- select statement is only one
  allowed here
  SELECT ...
```

Figure 1 - Syntax for an Inline Function

As you can see above, an Inline function takes in parameters, the parameters are manipulated in some way by a statement and a table is returned.

Lastly, a **multi-statement** table-valued function (MSTVF) expands on the idea of an inline statement by allowing a function to contain (as the name suggests) multiple-statements within it. It returns a result-set table like an inline function, but only after some additional processing. Figure 2 below shows the syntax of a MSTVF:

```
CREATE FUNCTION fnName(

  -- can have 0, 1, 2 or more
  parameters
  @param1 datatype,
  @param2 datatype, ...
)

-- define table to return
RETURNS @TableName TABLE (
  Column1 datatype,
  Column2 datatype,
  ...
  Columnn datatype,
)
AS
BEGIN

  -- typically insert rows into this
  table

  -- eventually, return the results
  RETURN

END
```

Figure 2 - Syntax of a Multi-Statement Function

Analyzing the mock code in Figure 2, you can see additional statements added when compared to Figure 1. Also, the additional statements/processing start and end with a BEGIN/END block.

Summary

In conclusion, Module 7 introduces the idea of User-Defined Functions as functions that a user can create to process data in a specific way. They come in 3 flavors: Scalar, Inline and Multi-Statement. A Scalar function returns a single value, whereas Inline and Multi-Statement functions return tables. Inline and Multi-Statement functions are similar but differ in that an Inline function as a single statement, compared to a Multi-Statement function (which has many). All-in-all, UDF's provide the user with a powerful tool to manipulate and process database data.