Raul Guajardo

03/03/2021

Foundations of Databases and SQL Programming

Assignment07

[Raul’sGitHub](https://github.com/RaulGuajardoOrtega/DBFoundations)

# Functions

Introduction

In this paper, I will be discussing what SQL User Defined Functions (UDFs) are, what the advantages of using them are and what differences exist between Scalar, Inline, and Multi-Statement Functions.

What is a UDF?

A **User Defined Function** is a subroutine made of one or multiple SQL transaction statements. It can accept parameters and perform actions, such as complex calculations and return the result of that action as a value. The return value can either be a single scalar value or a result set.

UDFs come in handy when there are repetitive tasks, such as extracting specific Data to be analyzed in a company, or simply because we don’t want to be writing the same lines of code every time we would like to work with the database.

Some advantages of using UDFs are:

* They can be stored in the database for future reuse.

Once created, a UDF can be stored in the database, and used any number of times. They can also be modified as needed.

* They allow faster execution.

UDFs reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated executions.

* They can reduce network traffic.

Some operations that cannot be expressed in a single scalar expression can be expressed as a function. The function can then be used in the Where clause to reduce the numbers of rows that return.

What are the main differences between Scalar, Inline and Multi-Statement Functions?

Scalar functions help to simplify the use of code. For example, sometimes we will have complex calculations that appear in many of our queries. Instead of including the code for the formula in every query, we can create a scalar function that encapsulates the formula and uses it in each query.

User Defined scalar functions return a single data value of the type defined in the **Returns** Clause. Figure 1 shows the code to create a scalar function.

**CREATE** **FUNCTION** sales.udfNetSale(

@quantity INT,

@list\_price DEC(10,2),

@discount DEC(4,2)

)

**RETURNS** DEC(10,2)

**AS**

**BEGIN**

**RETURN** @quantity \* @list\_price \* (1 - @discount)

**END**;

Figure 1 shows the code to create a scalar function that will help us to calculate the net sale of any sales order.

There are some cases where we don’t need to perform complex calculations, but instead compare or search for data containing certain characteristics in our databases. Table-valued functions come in handy in those cases, being Inline table-valued function the best option for performance and simplicity, and multi statement table-functions a second best, only using it when strictly necessary.

Let’s begin with a basic explanation of an Inline table-valued function (Inline TVF). An Inline TVF is a UDF that returns data of a table type. The return type of a table-valued function is a table, therefore, it can be used just like we would use a table. Figure 2 shows the code to create a table valued function.

CREATE FUNCTION [dbo].[udfGetProductList]

(@SafetyStockLevel SMALLINT

)

RETURNS TABLE

AS

RETURN

(SELECT Product.ProductID,

        Product.Name,

        Product.ProductNumber

FROM Production.Product

WHERE SafetyStockLevel >= @SafetyStockLevel);

Figure 2 shows the code necessary to create an Inline table-valued function.

In the other hand, Multi-statement functions can have more than one **Select** statements and they also must have a Begin/End block called the function body in the code structure in order to work properly. Multi-statement functions allow to build and insert rows into the table variable that is returned once the query is executed. Figure 3 shows the code necessary to create a Multi-statement function.

CREATE FUNCTION MultiStatement\_TableValued\_Function\_Name

(

        @param1 DataType,

        @param2 DataType,

        @paramN DataType

)

RETURNS

@OutputTable TABLE

(

        @Column1 DataTypeForColumn1 ,

        @Column2 DataTypeForColumn2

)

AS

BEGIN

  --FunctionBody

RETURN

END

Figure 3 shows the code to create a Multi-statement function.

In the following figure (Figure 4) we can appreciate the different roads and steps necessary to create Scalar, Inline table and Multi-statement functions.

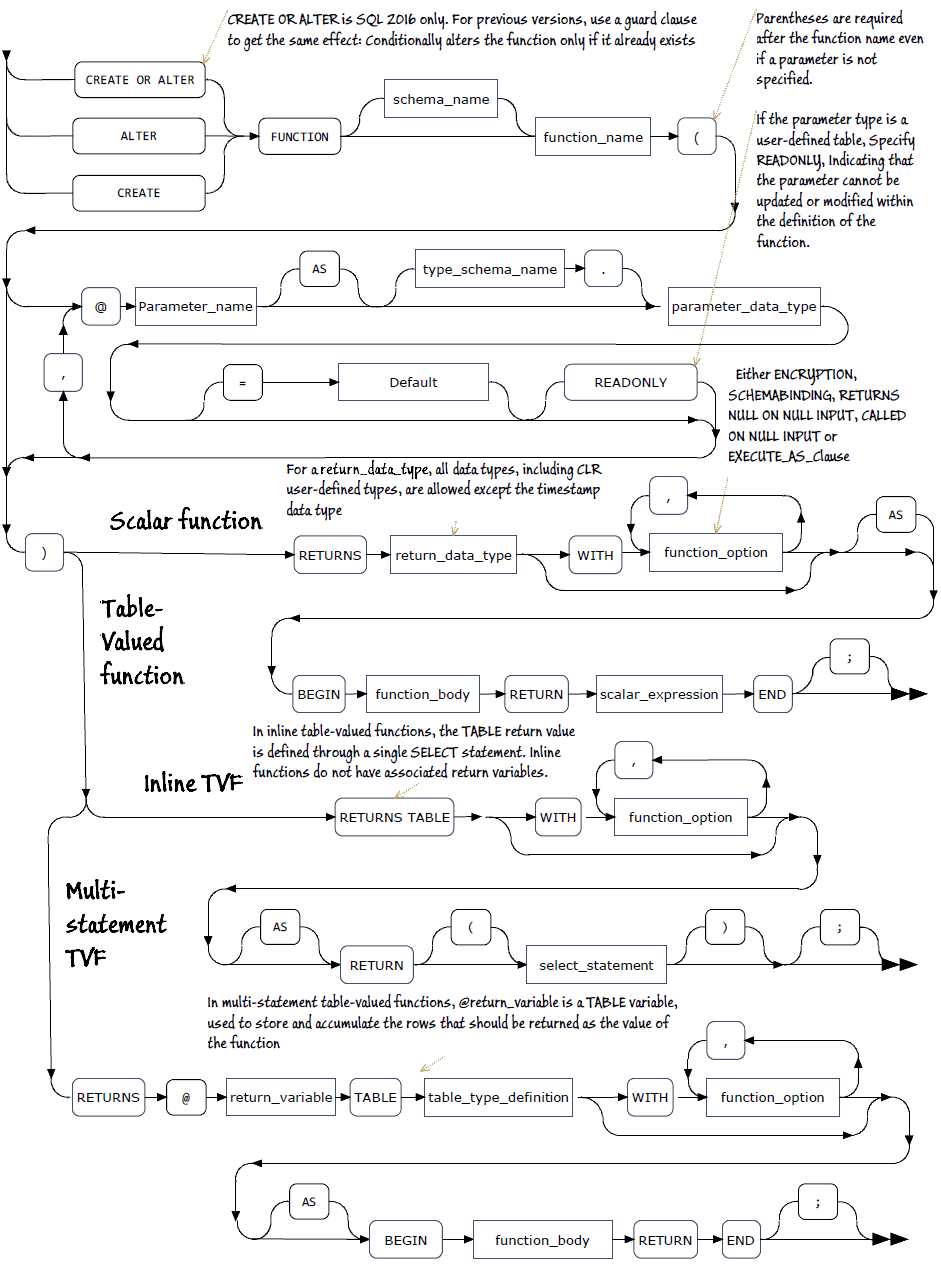


Figure 4 shows the Syntax diagram to create Scalar, Inline and Multi-statement functions. Source:

<https://www.red-gate.com/simple-talk/sql/t-sql-programming/sql-server-user-defined-functions/#:~:text=Inline%20functions%20do%20not%20have,such%20as%20modifying%20a%20table.>

Summary

It’s important to understand the implications of using UDFs in our code, since it can have significant impact in the performance of our databases. Whenever possible, we can first check for system functions to see if we can find a function that can achieve what we are trying to do. If the solution it’s not there, we then can go ahead and create a UDF that will return the data we are looking for and make our life easier by summoning every time we need it.