Alex Strait
29 November 2020
IT FDN 130 A - Foundations of Database Management
Assignment 07
Github Repository Link

SQL Functions

Introduction

Functions are somewhat more capable views. They can be created to return a table or single value, as opposed to the View's limitation of table only returns. Parameters can used when creating functions, allowing you pass in arguments to return individualized results in real time. As a result, each function will always require '()' syntax, even in instances where no parameters are passed. The below sections address when to use a user defined function (UDF) and the various subcategories you may come across.

SQL User Defined Functions

UDFs allow you to create your own custom functions to be called on within the same database as the creation. Relational Database Management Systems have countless pre-defined numeric, date, and string functions *(W3 Schools, https://www.w3schools.com/sql/sql_ref_sqlserver.asp, 2020)*, but there are limitations and if specific data manipulation is required, you will need a UDF. As a reminder, the *Create Function* statement must always be the first statement in a batch, requiring *go* just beforehand.

Scalar vs. Inline vs. Multi-Statement Functions

Multiple subcategories of UDFs exist. Inline and Multi-Statement table valued functions can often complete similar tasks in different ways. See below for various syntactical differences.

Scalar – Scalar functions *always* return a single value, and the schema must be included when identifying the function. Some existing examples include the aggregate functions: SUM, MIN, MAX, COUNT, and AVG.

Inline – A simple table-valued function that returns just a single set of rows. In this case, the attributes to display are determined in the Create Function statement at well as the definition of the parameter being used.

Multi-Statement – This sort of function can return results from multiple select statements in a single function, and as such is more involved when developing. The user defines a table variable in the Return block of the Create Function statement and by doing so, explicitly specifies the structure of the return table.

```
CREATE FUNCTION fnNameOfFunction(

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

)
RETURNS TABLE
AS
RETURN

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

```
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
```

Figures 1 & 2 – Example syntax for Inline Table-Valued Function (left) and Multi-Statement Table Valued Function *(Wise Owl, https://www.wiseowl.co.uk/blog/s347/table-valued-functions.htm, 2020)*

Conclusions

Functions do have their limitations - When reporting data through analysis, it is important to note that Excel and other applications often cannot import information from database functions. Instead, you can wrap a UDF into a view and import the view into these applications OR simply utilize a View with Where clauses as required to mimic function parameters. The best scenario for function creation is if a less-experienced user wants to pull various data in real time for multiple variable parameters.