Robert Boyd

8/22/2021

ITFDN130

Assignment 07

https://github.com/CaldwellAzvian/DBFoundations

**UDFs and Function Types in SQL**

**Introduction**

This week, we placed an emphasis on functions in SQL and how we can use them. In the following paper focusing on specifically user defined functions (UDFs), I will be explaining when we would want to use a SQL UDF and what the differences are between scalar, inline and multi-statement functions.

**UDFs and When We Use Them**

A UDF stands for a user defined function, which simply means that we are building our own function, beyond those that are already built into SQL that we can use. There are two kinds of values we can return: a single value or a table of values, so we could use a UDF when we want to return either of these. The powerful aspect of functions is that they can be passed parameters, so for these UDFs we can also define what kinds of parameters are received by the function. It is important to note however that in many reporting situations when we are returning a table, a view is very similar to a UDF, and if a parameter would be used to filter the output, it would be akin to using a where clause on the view. The ability to accept a parameter for this where clause however brings it value as something that can be done to easily query and filter the data if that function is used often. Another time we could use a UDF is for single values to be returned, which could be used to compress calculations into a smaller chunk, or even be used to create flexible calculations that would be dependent on some non-static value. One concrete application of this is guarding against data that would not fit within a time constraint, so for example if there was a database taking in time that an inventory was taken, one constraint you could add to the table is that the inventory could only be taken in the past and not in the future. A UDF within an alter table to adjust the table’s constraints could guard against future dated inventories should one be added.

**Types of Functions: Scalar, Inline, and Multi-Statement Functions**

The three types of functions we learned about this week were scalar, inline and multi-statement functions. A scalar function describes a function that returns only a single value. These are usually functions that perform formatting or calculations of some kind given parameters and returning the result. A simple example of a scalar function would be a function that performs a mathematical operation to two values such as addition for the sum, where you would pass it any two numbers and it would return the result of that calculation.

The other two kinds of functions fit under the umbrella of functions that return tables instead of a single value. An inline function returns a table of data, but it only returns a set of rows which makes it fairly simple and extremely similar to a view. A multi-statement function however is more complex and does some additional processing beyond simply returning rows. A multi-statement function could for example add rows into the returned table results, meaning that it could return some additional information based on the processing. It is important to note that it is inserting this information into the returned table, not actually affecting the tables in the database themselves. This ability to manipulate rows in the returned table will not affect permanent tables, and likewise you cannot use temporary tables in a UDF. In short, for a multi-statement function only the results being returned are affected and will not have an impact on the actual tables in the database.

**Summary**

Adding UDFs to our toolbox gives us the ability to effectively create condensed and clean code, while also expanding the flexibility of what we can do, return or calculate for a variety of purposes. Among those UDFs there are scalar functions for single returned values, inline functions for simple table reporting, and multi-statement functions that are capable of altering the selected table for some more advanced purpose before returning it.