Marcos Juki

May 24th, 2021

Foundations of Databases & SQL Programming

Assignment07

(<https://github.com/MarcosJuki/DBFoundations-Module07>)

**Views, Functions, and Stored Procedures**

**Introduction |**

This document will discuss when to use SQL Functions, their differences, and applications. Best practices and examples will also be displayed. The objective is to share knowledge with others in my workplace by gathering this knowledge.

**Explain when you would use a SQL UDF. |**

The importance behind User Defined Functions (or UDFs), like views, is the store frequently used queries in the database, avoid writing the same code repetitively, and creating an additional layer of protection to the data source, as permission, as well as accessibility are defined.

There are some other advantages to creating an UDF, and one I find frequently applicable to the workplace is, if code requires changes, if can simply be done in one spot and all applications that are using that function will reflect the alteration.

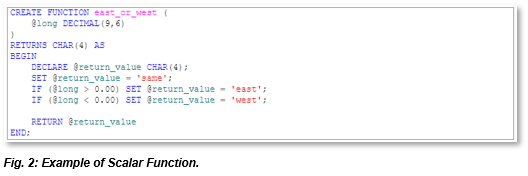
“The function takes parameters as input, does something with these input values (SQL statements); technically it will use values provided as parameters and combine them with other values (local variables) or database objects and then return the result of these combinations / calculations, and returns result of the calculation (RETURN value) with the previously defined type (RETURNS data\_type)” (\*)



**Explain the differences between Scalar, |**

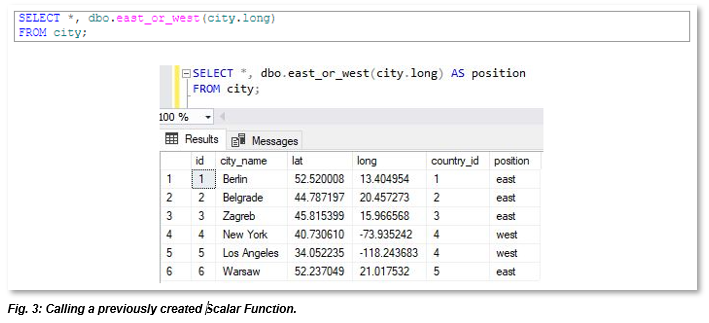
The Scalar user defined functions in SQL Server are very useful when you want to return a single value as the resultant. For example, total sales, or total investments, coordinates, etc.

In this example: “We want to list all cities and write down are they east or west when compared to London (longitude = 0). Cities east of London will have positive **city.long** values, while those west of London will have this value negative.



Our function takes a number as a parameter. The return value must be of the CHAR(4) type. The initial value (variable @return\_value) is initially set to *‘same’*. If the parameter (variable @long) is greater than 0, we’re *‘east’* from London, and if it’s less than 0, we’re *‘west’* of London. Notice that, in case of @long was 0, none of these two Ifs will change the value, so it will hold the initial value -> *‘same’*. This is really a simple function, but it’s a nice way to show what functions can do.

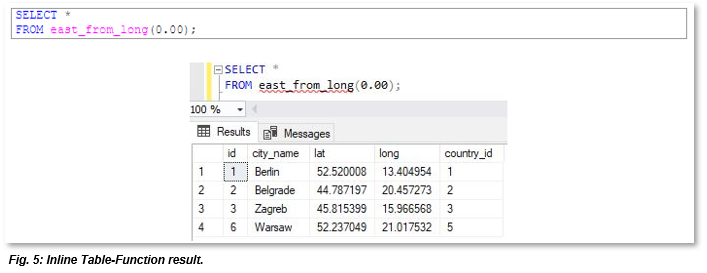
Note: You’ll call a function by simply using its name and providing the parameters needed. If the function is value-based, then you’ll be able to use this function at any place where you would use a number, string, etc.

The important thing to notice here is that we’ve used function as a “column” in our select query. We’ve passed parameter (**city.long** of the related row) and the function returned a result of the calculation. This is great because we’ve avoided writing complex calculations in a select query, and also, we can reuse this function later in any other query.” (\*)

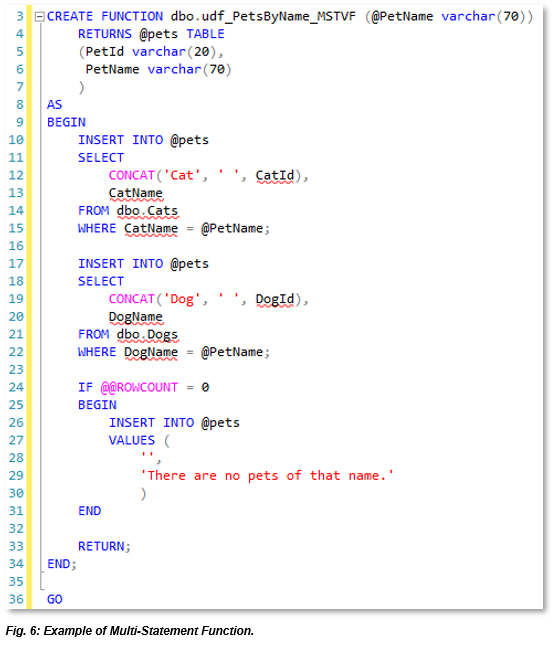
**Inline, |**

Inline functions allow it to return a table result set instead of just a single value. They essentially are a way for you to reuse a derived table query.

In this example: “We want to pass **long** as an argument and we’ll expect that function returns a table of all cities ‘east’ from the given parameter.” (\*)



(\*) (**SQL Shack**, <https://www.sqlshack.com/learn-sql-user-defined-functions/>, 2021, External Site)

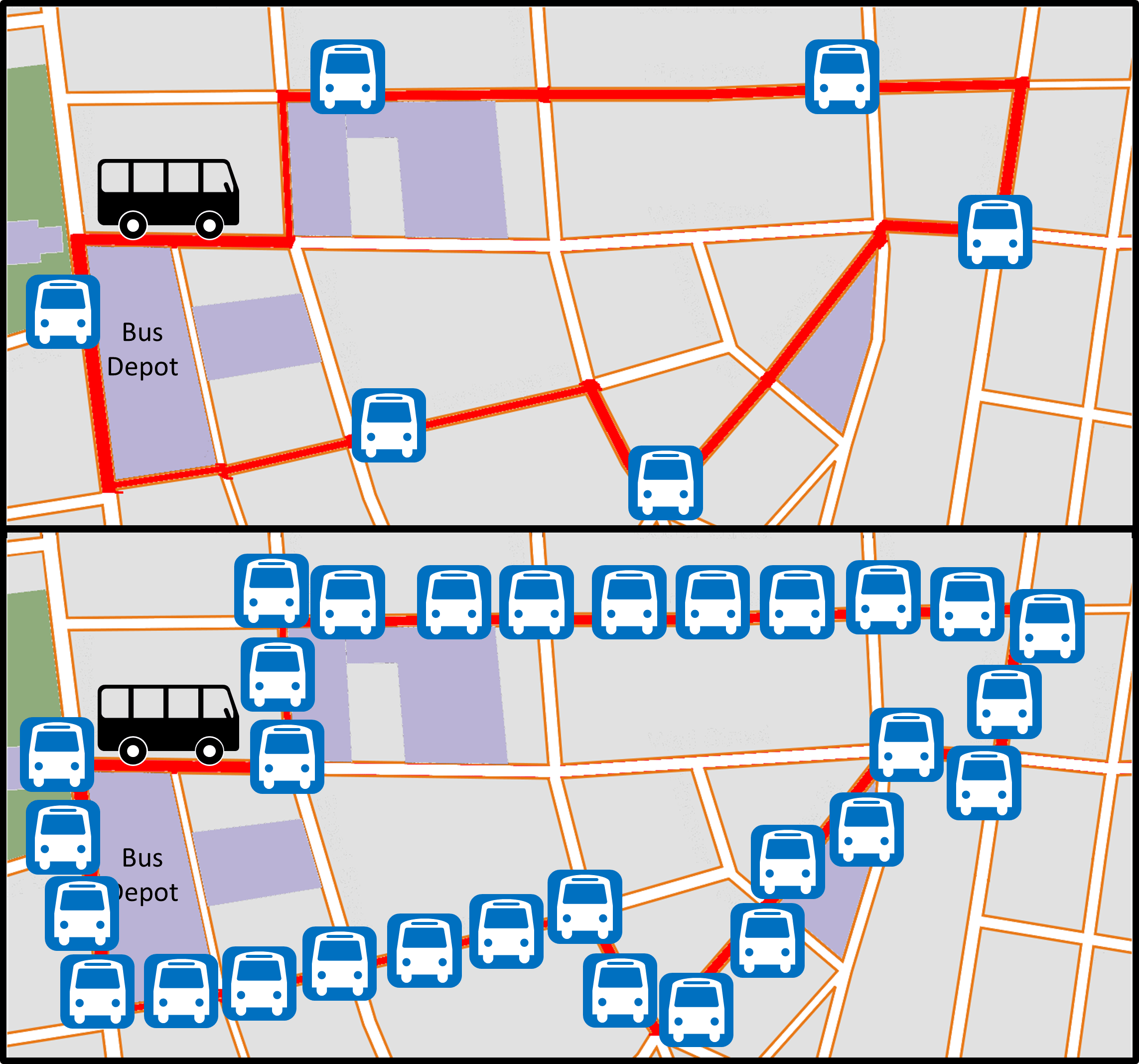
**and Multi-Statement Functions |**

“Multi-statement Table-Valued functions (sometimes referred to as MSTVFs) can consist of multiple statements, the results of which are stored in a return variable. You include the specifications of the return variable at the top of the function. This specifies the structure of the return table. In other words, you specify how many columns, their names, data types, etc.

The return table’s structure is defined at the start when I specify the @pets variable. The query results are inserted into the @pets variable.

In this case, the function requires a pet name be passed in as an argument. It then uses this argument in the queries in order to return the relevant data. Being a *multi*-statement table-valued function, I can include multiple statements in the function’s definition. ”

(**Database.Quide**, <https://database.guide/introduction-to-multi-statement-table-valued-functions-mstvf-in-sql-server/>, 2021, External Site)

**Summary |**

Here we have learned that Functions can cut down on the work involved when calling for complicated or extensive amount of data.

Imagine you have a delivery route with several stops on the way. The directions required to get to each location can have very specific instructions for each customer, and busines type. Now if you have them pre-programed on your GPS, with the required preestablished instructions, you can get to your destination much quicker. Same goes for functions, as they can be created to delivery results at a click.