Lisa Batten
Foundations of Databases and SQL Programming Class
November 29, 2021
GitHub Link: https://github.com/lsbadmadja/DBFoundations-Module07

Assignment 07
SQL User-Defined, Scalar, Inline, and Multi-Statement Functions

Introduction

In this article, I would like to talk about SQL User-Defined Functions (UDFs) and differences and similarities between Scalar, Inline, and Multi-Statement Functions.

SQL UDFs

An SQL User-Defined Function or UDF is a routine that accepts input parameters, performs an action, and returns a result. This result can be a single/scalar value or a set of values. A routine is a "programmable object that encapsulates codes to calculate a result or to execute activity". Other routines are stored procedures and triggers (Itzik 383). UDFs are usually used to (Itzik 384):

1. Allow users to save, re-use, and modify functions that they use repeatedly (modular programming).
2. Allow users to utilize and tailor system functions to their needs.
3. Allow users to include intensive logic and complex formulas in their codes.
4. Allow intensive logic or complex formulas to be executed faster and reduce network traffic through encapsulation of this intensive logic or complex formulas.

Scalar, Inline, and Multi-Statement Functions

In this section I would like to talk about similarities and differences between a Scalar, Inline, and Multi-Statement Function.

A Scalar Function is a function that accepts one or more parameters and returns a single value. A Table-Valued Function is a function that accepts one or more parameters and returns a set of values or a table. There are two Table-Valued Functions: Inline and Multi-Statement. The differences between the two are (Itzik 178, 179):

1. The Inline Function can only have one Select statement, but the Multi-Statement Function can have more than one statement.
2. The Inline Function cannot have Begin and End blocks, but the Multi-Statement Function can.
3. In the Inline Function, the structure of the table is defined by the Select statement, but, in the Multi-Statement Function, the structure of the result table is defined by users.
4. The Inline Function may be better in performance because it has only one Select statement.

Based on descriptions in above sections, we can conclude that similarities and differences between a Scalar, Inline, and Multi-Statement Functions are:

1. Similarities:
    1.1. All functions allow users to include intensive logic or complex formulas in their codes.
    1.2. All functions allow users to save, re-use, and modify functions that they use repeatedly.
    1.3. All functions allow users to utilize system functions and tailor them to their needs.
    1.4. All functions allow intensive logic or complex formulas to be executed faster and reduce network traffic through encapsulation of this intensive logic or formulas.
2. Differences:
    2.1. A Scalar Function returns a single value, but Inline and Multi-Statement Functions return a set of values.
    2.2. An Inline Function cannot have Begin and End blocks, but a Multi-Statement and Scalar Function can.
    2.3. An Inline Function can only have one Select statement, but a Multi-Statement Function can have more than one statement.
    2.4. In the Inline Function, the structure of the table is defined by the Select statement, but, in the Multi-Statement Function, the structure of the result table is defined by users.

2.5. An Inline Function may be better in performance because it has only one Select statement.

Summary

In this article, I talked about when to use SQL UDFs and differences and similarities between Scalar, Inline, and Multi-Statement Functions. To summarize:

1. An SQL UDF is a routine that accepts input parameters, performs an action, and returns a result. This result can be a single/scalar value or a set of values.
2. A routine is a "programmable object that encapsulates codes to calculate a result or to execute activity". UDFs, stored procedures, and triggers are SQL routines.
3. UDFs are usually used to:
   3.1. Allow users to save, re-use, and modify functions that they use repeatedly (modular programming).
   3.2. Allow users to utilize and tailor system functions to their needs.
   3.3. Allow users to include intensive logic and complex formulas in their codes.
   3.4. Allow intensive logic or complex formulas to be executed faster and reduce network traffic through encapsulation of this intensive logic or complex formulas.
4. A Scalar Function is a function that accepts one or more parameters and returns a single value.
5. A Table-Valued Function is a function that accepts one or more parameters and returns a set of values or a table. Two Table-Valued Functions are Inline and Multi-Statement.
6. Similarities and differences between a Scalar, Inline, and Multi-Statement Function are:
   6.1.Similarities:
      6.1.1. All functions allow users to include intensive logic or complex formulas in their codes.
      6.1.2. All functions allow users to save, re-use, and modify functions that they use repeatedly.
      6.1.3. All functions allow users to utilize system functions and tailor them to their needs.

6.1.4. All functions allow intensive logic or complex formulas to be executed faster and reduce network traffic through encapsulation of this intensive logic or formulas.

6.2.Differences:

6.2.1. A Scalar Function returns a single value, but Inline and Multi-Statement Functions return a set of values.

6.2.2. An Inline Function cannot have Begin and End blocks, but a Multi-Statement and Scalar Function can.

6.2.3. An Inline Function can only have one Select statement, but a Multi-Statement Function can have more than one statement.

6.2.4. In the Inline Function, the structure of the table is defined by the Select statement, but, in the Multi-Statement Function, the structure of the result table is defined by users.

6.2.5. An Inline Function may be better in performance because it has only one Select statement.

References

1. Ben-Gan, Itzik. "Table expressions". T-SQL Fundamentals. 3rd ed., Microsoft Press, 2016, pp. 161-192.
2. Ben-Gan, Itzik. "Programmable objects". T-SQL Fundamentals. 3rd ed., Microsoft Press, 2016, pp. 361-413.