SQL UDF

Name: Sonakshi
Date: 18th Dec 2021
Course: Foundations of Database and SQL Programming
Assignment: Assignment 7

Introduction

The document discusses User Defined Functions or UDF specially the aspect of when it is used and the different types of UDF that SQL provides.

When to Use SQL UDF

At a high level there are two kinds of SQL UDFs which are used to achieve different purposes:

1. **UDF returning scalar value:** A scalar UDF returns a single value as an expression. A scalar UDF can be used in the context of either checking constraints or for processing multiple or a single column values/value to get an inferred value.
2. **UDF returning table:** This type of UDF can be used to create a parameterized view, where we filter the data based on a passed set of parameters.

Difference between Scalar, Inline and Multi-Statement Functions

Scalar Function: A Scalar UDF always returns a single (scalar) value. A Scalar UDF can accept 0 to many input parameters and will return a single value. Here is an example of scalar function which takes two input parameters and returns the multiplied value:

```
Create Function dbo.MultiplyValues(@Value1 Float, @Value2 Float)
 Returns Float
 As
  Begin
   Return(Select @Value1 * @Value2);
  End
go
```

Inline Function: An inline table valued function returns a table variable instead of scalar value. The function body in case of inline function contains only one single SELECT statement. Here is an example:

```
Create Function dbo.GetEmployeeDetailsByDept(@DeptName Varchar(50))
 Returns Table
 AS
   Return (
     Select * from dbo.[EmployeeDetails]
     Where DeptName = @DeptName
   );
GO
```

Multi-Statement Function: A multi-statement function also returns a table and is different from Inline function by only the fact that it can have multiple SELECT statements inside the body as opposed to only one in inline function. Here is an example:

```
Create Function dbo.GetSalesByCustumerRegion(@pRegionName Varchar(50))
 RETURNS @vSalesByRegion Table
```

```
   (
        CountryName Varchar (50),
        GrossSales Float
   )
 AS
 BEGIN
  INSERT INTO @vSalesByRegion
  SELECT S1.Country, GrossSales = Sum(S1.GrossSales)
  FROM(
     SELECT
          R.[Region_Name]
         ,R.[Country]
         ,[Units_Sold]
         ,[Gross_Sales]
     FROM Sales S
     INNER JOIN
       Region R
     ON S.[SalesRegionId] = R.[RegionId]
     WHERE R.RegionName] = @pRegionName
   ) S1
   GROUP BY S1.Country

  IF @@ROWCOUNT = 0
  BEGIN
  INSERT INTO @vSalesByRegion
    VALUES ('No Country Found',0)
  END

RETURN
END
GO
```

Summary

In this document we looked at different types of UDF that we can define in SQL and their usages.