

Deepika Tripathi

May 30 ,2022

SQL Functions

Assignment07

<Git Hub URL>

SQL Functions

Introduction

SQL functions are simply sub-programs, which are commonly used and re-used throughout SQL database applications for processing or manipulating data. Basically, it is a **set of SQL statements that accept only input parameters, perform actions and return the result**. The function can return only a single value or a table. We can't use a function to Insert, Update, Delete records in the database table(s).

SQL Function

While a view is nothing more than a SQL statement that is stored in the database with an associated name. A function on the other hand returns a single value or a single result set after accepting input parameters.

Functions **foster code reusability**. If there is need to repeatedly write large SQL scripts to perform the same task, then we can create a function that performs that task. Next time instead of rewriting the SQL, specific function can be called.

The basic CREATE FUNCTION syntax is as follows –

```
Create function dbo.fProductInventoriesWithPreviousMonthCountsWithKPIs (@KPI int)
Returns Table
as

Return (Select Top 100000 PIK.ProductName, PIK.InventoryDate, PIK.Count,PIK.PreviousMonthCount, PIK.KPI
        from vProductInventoriesWithPreviousMonthCountsWithKPIs as PIK
        Where PIK.KPI= @KPI
        Order By PIK.ProductName, PIK.InventoryDate);
go
```

Fig1: Basic syntax of SQL function

View And Functions

In addition to SQL Server's built-in functions, you can create custom functions. These are often called User Defined Functions or just UDFs. There are two basic types of functions; functions that return a table of values and functions that return a single value.

- Functions and Views are similar as both can return a table of values.
- Unlike views, functions can use parameters to change the results of the query
- Unlike views, you can create UDFs to return a single (scalar) value as an expression.

Type of SQL functions

There are three types of user-defined functions in SQL Server:

- Scalar Functions (Returns A Single Value)
- Inline Table Valued Functions (Contains a single TSQL statement and returns a Table Set)
- Multi-Statement Table Valued Functions (Contains multiple TSQL statements and returns Table Set)

Difference between different types of functions

Scalar Functions: A scalar function accepts any number of parameters and returns one value. The term scalar differentiates a single, "flat" value from more complex structured values, such as arrays or result sets. This pattern is much like that of traditional functions written in common programming language.

```
SQL Copy

CREATE FUNCTION dbo.ISOweek (@DATE datetime)
RETURNS int
WITH EXECUTE AS CALLER
AS
BEGIN
    DECLARE @ISOweek int;
    SET @ISOweek= DATEPART(wk,@DATE)+1
        -DATEPART(wk,CAST(DATEPART(yy,@DATE) as CHAR(4))+ '0104');
    --Special cases: Jan 1-3 may belong to the previous year
    IF (@ISOweek=0)
        SET @ISOweek=dbo.ISOweek(CAST(DATEPART(yy,@DATE)-1
            AS CHAR(4))+ '12'+ CAST(24+DATEPART(DAY,@DATE) AS CHAR(2)))+1;
    --Special case: Dec 29-31 may belong to the next year
    IF ((DATEPART(mm,@DATE)=12) AND
        ((DATEPART(dd,@DATE)-DATEPART(dw,@DATE))>= 28))
        SET @ISOweek=1;
    RETURN(@ISOweek);
END;
GO
SET DATEFIRST 1;
SELECT dbo.ISOweek(CONVERT(DATETIME, '12/26/2004',101)) AS 'ISO Week';
```


Here is the result set.

```
Copy

ISO Week
-----
52
```

Fig2: Using a scalar-valued user-defined function that calculates the ISO week

Inline Table-Valued Functions: This type of function returns a result set, much like a view. However, unlike a view, functions can accept parameters. The inline function's syntax is quite simple. In the function definition, the return type is set to a table. A return statement is used with a select query in parenthesis.

SQL 

```
CREATE FUNCTION Sales.ufn_SalesByStore (@storeid int)
RETURNS TABLE
AS
RETURN
(
    SELECT P.ProductID, P.Name, SUM(SD.LineTotal) AS 'Total'
    FROM Production.Product AS P
    JOIN Sales.SalesOrderDetail AS SD ON SD.ProductID = P.ProductID
    JOIN Sales.SalesOrderHeader AS SH ON SH.SalesOrderID = SD.SalesOrderID
    JOIN Sales.Customer AS C ON SH.CustomerID = C.CustomerID
    WHERE C.StoreID = @storeid
    GROUP BY P.ProductID, P.Name
);
GO
```

To invoke the function, run this query.

SQL 

```
SELECT * FROM Sales.ufn_SalesByStore (602);
```

Fig3: Creating an inline table-valued function

Multi-Statement Table-Valued Functions: Multi-Statement functions can be used to do some unique operations outside the context of a standard SELECT statement. This type of function returns a table-type result set, but the table is explicitly constructed in script. This can be used to accomplish one of two things: either to process some unique business logic by assembling a virtual table on the fly, or to duplicate the functionality of an inline function in a more verbose and compiled way. In short, if you need to select records from an existing result set, use an inline table-valued function.

```
CREATE FUNCTION dbo.ufn_FindReports (@InEmpID INTEGER)
RETURNS @retFindReports TABLE
(
    EmployeeID int primary key NOT NULL,
    FirstName nvarchar(255) NOT NULL,
    LastName nvarchar(255) NOT NULL,
    JobTitle nvarchar(50) NOT NULL,
    RecursionLevel int NOT NULL
)
--Returns a result set that lists all the employees who report to the
--specific employee directly or indirectly.*/
AS
BEGIN
WITH EMP_cte(EmployeeID, OrganizationNode, FirstName, LastName, JobTitle,
RecursionLevel) -- CTE name and columns
AS (
    -- Get the initial list of Employees for Manager n
    SELECT e.BusinessEntityID, OrganizationNode = ISNULL(e.OrganizationNode,
CAST('/' AS hierarchyid))
    , p.FirstName, p.LastName, e.JobTitle, 0
    FROM HumanResources.Employee e
        INNER JOIN Person.Person p
        ON p.BusinessEntityID = e.BusinessEntityID
    WHERE e.BusinessEntityID = @InEmpID
    UNION ALL
    -- Join recursive member to anchor
    SELECT e.BusinessEntityID, e.OrganizationNode, p.FirstName, p.LastName,
e.JobTitle, RecursionLevel + 1
    FROM HumanResources.Employee e
        INNER JOIN EMP_cte
        ON e.OrganizationNode.GetAncestor(1) = EMP_cte.OrganizationNode
        INNER JOIN Person.Person p
        ON p.BusinessEntityID = e.BusinessEntityID
    )
-- copy the required columns to the result of the function
INSERT @retFindReports
SELECT EmployeeID, FirstName, LastName, JobTitle, RecursionLevel
FROM EMP_cte
RETURN
END;
GO
-- Example invocation
SELECT EmployeeID, FirstName, LastName, JobTitle, RecursionLevel
FROM dbo.ufn FindReports(1);

GO
```

Fig4: Creating a multi-statement table-valued function

Example screenshots are captured from: [CREATE FUNCTION \(Transact-SQL\) - SQL Server | Microsoft Docs](#)

Summary

SQL Views, Function & Stored Procedures are used to retrieve data from the database. SQL functions are used to return results in 3 different formats after taking inputs from the caller. SQL functions are most helpful in scenarios where a piece of code has to be used in multiple places or there are complex manipulations of the data need to happen without updating the data in table.