## [MS SQL](https://www.databasejournal.com/features/mssql/)

Posted May 7, 2004

# User Defined Functions in Microsoft SQL Server 2000

This article will explore the uses, restrictions and benefits of User Defined Functions in Microsoft SQL Server 2000

### Introduction

User Defined Functions are compact pieces of Transact SQL code, which can accept parameters, and return either a value, or a table. They are saved as individual work units, and are created using standard SQL commands. Data transformation and reference value retrieval are common uses for functions. LEFT, the built in function for getting the left part of a string, and GETDATE, used for obtaining the current date and time, are two examples of function use. User Defined Functions enable the developer or DBA to create functions of their own, and save them inside SQL Server.

### Advantages of User Defined Functions

Before SQL 2000, User Defined Functions (UDFs), were not available. Stored Procedures were often used in their place. When advantages or disadvantages of User Defined Functions are discussed, the comparison is usually to Stored Procedures.

One of the advantages of User Defined Functions over Stored Procedures, is the fact that a UDF can be used in a Select, Where, or Case statement. They also can be used to create joins. In addition, User Defined Functions are simpler to invoke than Stored Procedures from inside another SQL statement.

### Disadvantages of User Defined Functions

User Defined Functions cannot be used to modify base table information. The DML statements INSERT, UPDATE, and DELETE cannot be used on base tables. Another disadvantage is that SQL functions that return non-deterministic values are not allowed to be called from inside User Defined Functions. GETDATE is an example of a non-deterministic function. Every time the function is called, a different value is returned. Therefore, GETDATE cannot be called from inside a UDF you create.

### Types of User Defined Functions

There are three different types of User Defined Functions. Each type refers to the data being returned by the function. Scalar functions return a single value. In Line Table functions return a single table variable that was created by a select statement. The final UDF is a Multi-statement Table Function. This function returns a table variable whose structure was created by hand, similar to a Create Table statement. It is useful when complex data manipulation inside the function is required.

### Scalar UDFs

Our first User Defined Function will accept a date time, and return only the date portion. Scalar functions return a value. From inside Query Analyzer, enter:

CREATE FUNCTION dbo.DateOnly(@InDateTime datetime)

RETURNS varchar(10)

AS

BEGIN

DECLARE @MyOutput varchar(10)

SET @MyOutput = CONVERT(varchar(10),@InDateTime,101)

RETURN @MyOutput

END

To call our function, execute: SELECT dbo.DateOnly(GETDATE())

Notice the User Defined Function must be prefaced with the owner name, DBO in this case. In addition, GETDATE can be used as the input parameter, but could not be used inside the function itself. Other built in SQL functions that cannot be used inside a User Defined Function include: RAND, NEWID, @@CONNCECTIONS, @@TIMETICKS, and @@PACK\_SENT. Any built in function that is non-deterministic.

The statement begins by supplying a function name and input parameter list. In this case, a date time value will be passed in. The next line defines the type of data the UDF will return. Between the BEGIN and END block is the statement code. Declaring the output variable was for clarity only. This function should be shortened to:

CREATE FUNCTION testDateOnly(@InDateTime datetime)

RETURNS varchar(10)

AS

BEGIN

RETURN CONVERT(varchar(10),@InDateTime,101)

END

### Inline Table UDFs

These User Defined Functions return a table variable that was created by a single select statement. Almost like a simply constructed non-updatable view, but having the benefit of accepting input parameters.

This next function looks all the employees in the pubs database that start with a letter that is passed in as a parameter. In Query Analyzer, enter and run:

USE pubs

GO

CREATE FUNCTION dbo.LookByFName(@FirstLetter char(1))

RETURNS TABLE

AS

RETURN SELECT \*

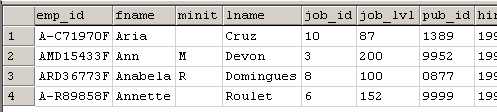
FROM employee

WHERE LEFT(fname, 1) = @FirstLetter

To use the new function, enter:

SELECT \* FROM dbo.LookByFName('A')

All the rows having a first name starting with A were returned.



The return is a Table Variable, not to be confused with a temporary table. Table variables are new in SQL 2000. They are a special data type whose scope is limited to the process that declared it. Table variables are stated to have performance benefits over temporary tables. None of my personal testing has found this result though.

**User Defined Functions in Microsoft SQL Server 2000 - Page 2**

**Multi Statement UDFs**

Multi Statement User Defined Functions are very similar to Stored Procedures. They both allow complex logic to take place inside the function. There are a number of restrictions unique to functions though. The Multi Statement UDF will always return a table variable--and only one table variable. There is no way to return multiple result sets. In addition, a User Defined Function cannot call a Stored Procedure from inside itself. They also cannot execute dynamic SQL. Remember also, that UDFs cannot use non-deterministic built in functions. So GETDATE and RAND cannot be used. Error handling is restricted. RAISERROR and @@ERROR are invalid from inside User Defined Functions. Like other programming languages, the purpose of a User Defined Function is to create a stand-alone code module to be reused over and over by the global application.

For a Multi Statement test, we will create a modified version of the LookByFName function. This new function will accept the same input parameter. But rather than return a table from a simple select, a specific table will be created, and data in it will be manipulated prior to the return:

CREATE FUNCTION dbo.multi\_test(@FirstLetter char(1))

RETURNS @Result TABLE

(

fname varchar(20),

hire\_date datetime,

on\_probation char(1)

)

AS

BEGIN

INSERT INTO @Result

(fname, hire\_date)

SELECT fname, hire\_date

FROM employee

WHERE LEFT(fname, 1) = @FirstLetter

UPDATE @Result

SET on\_probation = 'N'

UPDATE @Result

SET on\_probation = 'Y'

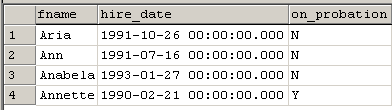
WHERE hire\_date < '01/01/1991'

RETURN

END

To use the new function, execute:

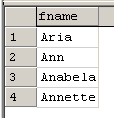
SELECT \* FROM dbo.multi\_test('A')



With the new Multi Statement Function, we can manipulate data like a Stored Procedure, but use it in statement areas like a View.

For example, only specific columns can be returned.

SELECT fname FROM dbo.multi\_test('A')

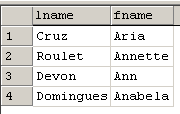


The function can also be joined like a view:

SELECT e.lname, f.fname

FROM employee e INNER JOIN dbo.multi\_test('A') f ON

e.fname = f.fname



**Conclusion**

User Defined Functions offer an excellent way to work with code snippets. The main requirement is that the function be self-contained. Not being able to use non-deterministic built in functions is a problem, but if it can be worked around, UDFs will provide you with a programming plus.

**»** [See All Articles by Columnist **Don Schlichting**](https://www.databasejournal.com/article.php/3085181)