Recursive Scalar Function in T-SQL

In my Common Table Expressions presentation the topic of recursion often comes up, but for scalar functions in T-SQL, it might not be as common.
This article has been written to show how a scalar function in SQL Server can call itself, thus being considered recursive.

The example uses a recursive scalar function to calculate the Fibonacci sequence.

What is the Fibonacci Sequence

The Fibonacci Sequence is a classic example that is used to demonstrate recursion.

By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

For instance, the following are Fibonacci Numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817, 39088169, 63245986, 102334155, 165580141, 267914296, 433494437, 701408733 …

Fibonacci Sequence as a Computer Science Challenge

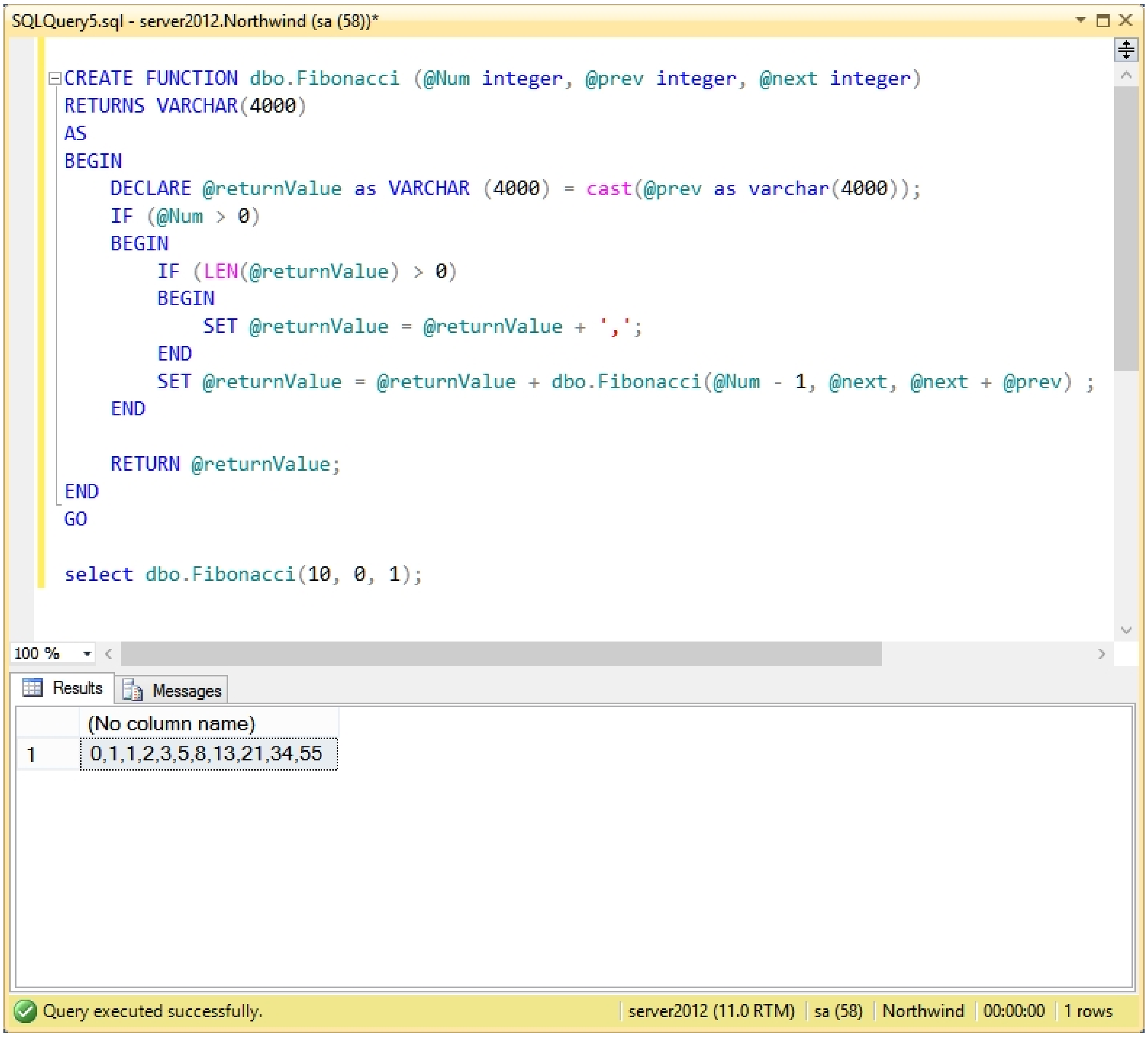
Often times calculating the **Fibonacci Sequence** is used as a computer science puzzle, or programming interview question to see if you understand recursion. It is very simple to do in any programming language that supports recursion.

What is Recursion

Recursion is a programming concept where a function, procedure or section of code calls itself. For instance in the following T-SQL example, the scalar function Fibonacci calculates the Fibonacci sequence using recursion, by calling itself. This is accomplished by calling the function name inside of the body of the function.

To call the function you simply include in a select statement, with the first parameter being the number of Fibonacci numbers to calculate, and the second and third parameters are always 0 and 1. The second and third parameters are set to 0 and 1 to prime the recursive function, and are used internally to pass the recursive call the current and previous values.

Which produces the following output using SSMS:



```sql
CREATE FUNCTION dbo.Fibonacci (@Num integer, @prev integer, @next integer)
RETURNS VARCHAR(4000)
AS
BEGIN
    DECLARE @returnValue as VARCHAR (4000) = cast(@prev as varchar(4000));
    IF (@Num > 0)
    BEGIN
        IF (LEN(@returnValue) > 0)
        BEGIN
            SET @returnValue = @returnValue + ',';
        END
        SET @returnValue = @returnValue + dbo.Fibonacci(@Num - 1, @next, @next + @prev) ;
    END

    RETURN @returnValue;
END
GO

select dbo.Fibonacci(10, 0, 1);
```

	(No column name)
1	0,1,1,2,3,5,8,13,21,34,55

Query executed successfully. | server2012 (11.0 RTM) | sa (58) | Northwind | 00:00:00 | 1 rows

For more details on the CTE version of Fibonaci take a look at my earlier post.

Enjoy!