CTE to Split a String in a function SplitString

Back in 2012 when I was writing my Common Table Expressions book, I came up with the following CTE in a function to split a string into rows.

Original Post: http://stevestedman.com/2012/04/using-a-cte-to-split-a-string-into-rows/

At the time, I thought it was pretty handy function, I ended up including it as an example in my CTE book, and I have used it dozens of times over the last couple of years.

SQL Server 2016

In SQL Server 2016 Microsoft has introduced a built in function called STRING_SPLIT which is very similar to this example. If you are using SQL Server 2016 you may want to consider the STRING_SPLIT function instead. The examples shown here work great and have been tested on SQL Server 2005, SQL Server 2008, SQL Server 2008R2, SQL Server 2012, SQL Server 2014 and SQL Server 2016. If Microsoft has done it right in SQL Server 2016, I would expect that their version of SPLIT_STRING will have better performance.

Performance with splitting strings

This last week I was working with a client who had a performance issue on a SSRS report that took over an hour to run, the person working on the report did an excellent job tracking it down. In SSRS when you have the option to select multiple items from a list with a check box, the id's from those items are passed in to the report as a string that is comma separated. Something like this:

"12334, 34234, 2342332, 234234, 23423"

In order to use that in a query it needs to be split out into something that can easily be used. The way I have typically seen this done is with a split function like the one above. They had a split function but it didn't use a CTE, instead the function used a loop to iterate through the list and break it up into rows.

Which took over an hour to run. He had also through some testing with hard coded values determined that the following ran in just a second.

In this case Table1 and Table2 were both very large tables, with some great indexes.

Why did it take so long to run with the originalSplitFunction?

Here is what I knew as we started looking into it:

- The originalSplitFunction when included in the query took over an hour to run.
- The originalSplitFunction contained multiple statements with a loop and some work to fill up the result table.
- The SQL Server query optimizer treats multiple statement table valued functions as a black box when analyzing the plan. The optimizer doesn't know anything about how the function works, or how many results will be returned.
- The SQL Server query optimizer is able to use single statement table valued functions when analyzing the plan, therefore if we have a string split function that is a single statement, we should be able to end up with a better plan.
- The use of the originalSplitFunction is causing a full table scan on a very large table.

The new and improved function

That's the point that I remembered my split function using a CTE that only has a single statement. We grabbed the split function from my blog, and with some minor adjustments to handle VARCHAR(MAX), we ended up with the following function.

When we replaced the originalSplitFunction with the SplitString function the query ran in about a second. From 1 hour down to a second run time for a report. I hope someone notices that improvement.

Related Links:

- Original Post: http://stevestedman.com/2012/04/using-a-cte-to-split-a-string-into-rows/