SQL Server Memory Hog Query

The article could be named, "How to use up all of your SQL Server available memory with a single CTE query." Another name for the article could just be "SQL Server Bug Report" depending on how you look at it.

When presenting unleashing Common Table Expressions at SQL Saturday a while back, I was asked a couple of great questions that I didn't know the answer to. So I did the research and tracked it down:

1. How many levels of recursion can you have in a CTE?

2. How many levels of nesting can you have in a CTE?

So I started doing the research and doing some testing to figure it out.

How many levels of recursion can you have in a CTE?

This is the easier one to answer between the two questions. The answer is, you can have more levels of recursion that you would ever need for standard recursion. I have tested CTEs with up to 1 million levels of recursion, and the have performed pretty well. If you are writing queries that need more than 1 million levels of recursion, perhaps you should take a look at a different approach.

How many levels of nesting can you have in a CTE?

This is where it gets really interesting. SQL Server does a really good job with overall performance on CTEs, but where it completely breaks down is on deep nested CTE queries. By deep nested I mean more than a thousand or two thousand. To answer the question of how many levels can you have, I would answer this as you can nest more than you would ever really need. If you really need more than a thousand levels of nesting in a CTE, you might want to rethink your approach.

On SQL Server 2005 the limit is 255. But in SQL Server 2008 and newer this limit was extended, and appears to not have a fixed limit, rather the limit is based on the amount of memory available for the query to use.

So to build the crazy CTE, I used excel to build out several thousands of rows of nested CTEs. One calling another, calling the next, and so on. What I found was that this was a very easy way to use up almost all the memory on your database. DO NOT TRY THIS ON A PRODUCTION DATABASE. The reason that DBAs and developers have test or development databases is to play around with things that may be dangerous on a production server. This is one that you could try on a test server.

The other interesting thing that the query does when it uses up all the memory, somehow it dumps some of the connections that are currently active on the SQL Server at that point.

Watch the video to find out how a nested CTE query could use up all of the memory on your SQL Server.

http://www.youtube.com/embed/KOvqUjE6714

If anyone has a test server with 512GB, or 1TB of available, memory, I would love to see how this type of query performs. Give it a try and let me know.

You can download the script NestedCTE.zip here. The zip file contains 2 files, one called CTE.sql, and one called CTE2500.sql. The 2500 version is the one that I used in the demo, and the CTE.sql has 32767 nestings. I would love to see the bigger on run on a server with a TB or more of memory to see how it performs.

Here are a few other links to CTE related posts on my site.

- Using a CTE in a Function to Split Up a Query String
- Using a CTE to Split a String Into Rows
- Can we use CTEs use in SSRS?
- CTE Hierarchy compared to the alternative
- Calculating Factorials with a Recursive CTE
- CTE – With An Insert Statement
- Fibonacci Sequence
- Generating a Tree Path with a CTE
- CTE Query Performance
- Multiple CTE's in a single Query
- Recursive CTE's
- Common Table Expressions Page
- A Fun Thanksgiving Day Query.
- Common Table Expressions