DBCC ShrinkDatabase

Being day 24 of the DBCC Command month at SteveStedman.com, today's featured DBCC Command is DBCC SHRINKDATABASE.

When I first heard about DBCC Shrink Database (many years ago), I immediately thought "what a great feature, run this every night when the system load is low, and the database will be smaller, perform better, and all around be in better shape". Boy was I wrong.

To 'DBCC SHRINKDATABASE' or Not To 'DBCC SHRINKDATABASE': What's the question

If you read Microsoft Books Online, there is a lot of great info on all the benefits of shrinking your database, and hidden down in the best practices section one line about how "A shrink operation does not preserve the fragmentation state of indexes in the database".

So why not just shrink the database every day like I had attempted so many years ago. The problem is index fragmentation, which is a pretty common problem on many SQL Servers. Index fragmentation is such a performance issue that the other obsolete DBCC commands DBCC IndexDefrag and DBCC DBReIndex were created, and later replaced with ALTER INDEX options for rebuilding and reorganizing

Is it a good idea to run DBCC SHRINKDATABASE regularly?

Download the sample file ShrinkSample.

This article and samples apply to SQL Server 2005, 2008, 2008R2, SQL 2012, and SQL Server 2014.

This really depends on a number of factors, but generally the answer is NO, it is not a good idea to run DBCC SHRINKDATABASE regularly.

For the purpose of this article, I am going to assume a couple of things:

1. You are concerned about database performance.
2. Over time your database is growing (which is probably why are are concerned about performance).
3. You want to do your best to improve the overall health of the database, not just fixing one thing.

Most DBAs who are not familiar with the issues around index fragmenation just set up maintenance

plans, and see SHRINKDATABASE as a nice maintenance plan to add. It must be good since it is going to make the database take up less space than it does now. This is the problem, although SHRINKDATABASE may give you a small file, the amount of index fragmentation is massive.

I have seen maintenance plans that first reorganize or rebuild all of the indexes, then call DBCC SHRINKDATABASE. This should be translated as the first reorganize all of the indexes, then they scramble them again.

Here is an example showing some new tables, with a clustered index on the largest, that are then fragmented, then REORGANIZED, then SHRINKDATABASE. You might find the results interesting.

To start with, I am going to create a new database, with two simple tables. One table uses a CHAR column and the other users VARCHAR. The reason for the CHAR column is to just take up extra space for the purpose of the demonstration. Each table will be filled with 10,000 rows holding text that is randomly generated with the NEWID() function and then cast to be a VARCHAR. For the purpose of demonstrating, that appeared to be a good way to fill up the table with some characters.

Now that we have some tables, lets take a look at the size of the database and the fragmentation on Table2. We will run thee

following two queries before after each of the commands following this.

The results of the two checks are shown below. You can see that the "Logical scan fragmentation" is 2.9% which is very good. You can also see that the data file is taking 80.0mb of disk space. Remember these numbers as they will be changing later.

Next we drop Table1 which will free up space at the beginning of the datafile. This is done to force Table2 to be moved when we run DBCC SHRINKDATABASE later.

The checks after dropping the table show that there is no change to the Table2 fragmentation, but free space in the datafile is now 78.38mb.

Next we shrink the database, then run the same 2 queries to check the size and the fragmentation.

The results show good news and bad news. The good news is that the filesize has been reduced from 80mb to just 1.88mb. The bad news shows that fragmentation is now 98.55%, which indicates that the index is not going to perform as optimal as it should. You can see the shrinkdatabase has succeeded just as expected, and if you didn't know where to look, you wouldn't know that the clustered index on Table2 is now very fragmented.

Imagine running DBCC SHRINKDATABASE every night on a large database with hundreds or thousands of tables. The effect would be that very quickly every table with a clustered index would end up at close to 100% fragmented. These heavily fragmented indexes will slow down queries and seriously impact performance.

To fix this fragmentation, you must REORGANIZE or REBUILD the index.
The standard recommendation is to REORGANIZE if the fragmentation is between 5% and 30%, and to REBUILD if it is more than 30% fragmented. This is a good recommendation if you are running on SQL Server Enterprise Edition with the ability to REBUILD indexes online, but with standard edition this is not available so the REORGANIZE will do the job.

Once we run this our check script shows that after the REORGANIZE the fragmentation has been reduced to 10.14%, which is a big improvement over the 98.55% it was at earlier.

Next we try the REBUILD.

Which reduces the fragmenation to 4.17%, but it increases the filesize to 34.88mb. This effectively is undoing a big part of the original DBCC SHRINKDATABASE.

Notes

You can REBUILD or REORGANIZE all of your indexes on the system at one time, but this is not recommended. The REBUILD or REORGANIZE of all of the indexes will impact performance while it is running, and it may cause excessive transaction logs to be generated.

After doing a REORGANIZE of an index, it is suggested that statistics be updated immediately after the REORGANIZE.

Summary

It is my opinion that DBCC SHRINKDATABASE should never be run on a production system that is growing over time. It may be necessary to shrink the database if a huge amount of data has been removed from the database, but there are other options besides shink in this case. After any DBCC SHRINKDATABASE, if you chose to use it, you will need to REBUILD or REORGANIZE all of your indexes.

Even if you never use DBCC SHRINKDATABASE your indexes will end up getting fragmented over time. My suggestion is to create a custom Maintenance Plan which finds the most fragmented indexes and REBUILD or REORGANIZE them over time. You could for instance create a stored procedure that finds and REORGANIZES the 4 or 5 indexes that are the most fragmented. This could be run a couple times per night during a slow time allowing your system to automatically find and fix any indexes that are too fragmented.

Related Posts:

Blog: Index Fragmentation

Blog: Index Fragmentation and SHRINKDATABASE

Notes:

For more information see TSQL Wiki DBCC shrinkdatabase.

DBCC Command month at SteveStedman.com is almost as much fun as realizing how fragmented your indexes are after running DBCC SHRINKDATABASE.