SQL Server DBCC Commands: Shrinking Notes

This chapter refers to shrinking your database files (mdf, or ndf files), not shrinking the log file. The log file is a completely different conversation, however, shrink database does shrink the log file.

Not shrinking your database is one of the more counter intuitive things out there. You might think that a smaller database is a good thing, however there are some negative side effects if you shrink your database regularly, or have the autoshrink option enabled. Side effects of shrinking your database include:

* Excessive I/O due to the shrink.

* Index fragmentation (most likely all of your indexes).

* Excessive I/O to defragment your indexes.

* After the shrink is complete, inserting or updating rows that require more space in your database will be slowed due to the time involved with growing your data file.

Think of it like this, let's say you run a toy store. Your store starts out at 100,000 square feet of space. That's the perfect amount of space for you to use before the holidays. Now imagine if after the holidays, you only need 10,000 square feet of space, so you remodel your toy store (with great expense) to be smaller, then when the next holiday season comes along you need to remodel your store again (back to 100,000 square feet) to make room for the holiday toy inventory. You can see how expensive this might get to be continuously remodeling your toy store. Now if things happen that permanently change the need for square footage at your store, it may make sense to do a onetime remodel, but it wouldn't make sense to regularly remodel to accommodate the space needed today.

SQL Server databases are very similar to the toy store example in that it doesn't make sense to shrink the size just because that space isn't being used today. There is a dangerous autoshrink option that you can turn on for your database which will regularly shrink the size of your database for you. Other dangerous alternatives are to have DBCC SHRINK DATABASE or SHRINKFILE as a scheduled job.

Let's not confuse purging obsolete data with shrinking the database. Think of the size of the database as a container, and the amount of space used in that container is made up of data, indexes, and other structures.

Purging obsolete data is good, this will reduce the size of your tables, your indexes, and will generally if accompanied by index defragmentation leading to better performance.

Shrinking the database is not so good. The shrink process makes the database file, or the container for all your data as small as it possibly can be, which leads to the performance issues outlined above.

Here is why I am suggesting to not shrink the database, there is very little to be gained by the shrink database command, other than freeing up some space on disk. This size of the backups are not determined by the size of the data file, but by the number of 8k data pages in that file that are being used. Deleting the data, and rebuilding indexes, will help speed up the backups, the checkdb time, and the statistics work. Having the data files around 10% to 20% larger than currently needed is a good practice, that extra free space will be used when new data gets added, and to be able to use existing free space in the data file is much faster that having to expand the data file.

Free space in the data file is totally ignored during the CheckDB process, during the backup process, and during the statistics and index rebuilding process, and most of all it is ignored when normal queries are being run.

Let's assume that my data file was once 1TB (1000gb) in size. I have purged old obsolete data, and the tables, and indexes are now about 100gb in size, or roughly 10% of the size of the entire data file. Should I shrink my database now?

This may make sense to shrink the database if most of the following are all true.

* My database is not going to grow back to the 1TB size anytime in the next year or two.

* I have enough time during off hours, or low load time to shrink the database, possibly 12 or more hours depending on the size, and performance of the server.

* I am tight on disk space, and need to regain space for some other purpose besides this database.

* If I shrink the database, I am planning on rebuilding or reorganizing all of my indexes.

* If after I shrink the database I am going to expand it a bit (10% to 20%) to accommodate growth.

* I understand that performance will be impaired while the database is shrinking and after the shrink until my indexes are rebuilt.

Log files are a completely different beast from data files. There are many occasions that it may be completely reasonable to shrink log files, for instance to reduce VLF counts by shrinking the log file, then growing it back in large chunks. Another example would be if your log backups had been failing for some reason that caused the log file to grow excessively, this would be a good reason to shrink the log file. That's a while different situation from shrinking the database files. Before considering shrinking the log files be sure to do your research and have a good understanding of

what is involved.

There are very few occasions that will justify shrinking your database, and these should be few and far between. I have never come across a situation where setting the automatic shrink, or regularly scheduled shrinking of the database was a good idea.

Don't shrink your database unless you have a really good understanding of the negative impacts, and you have a significant reason to do so. Often times if the reason to shrink your database is to regain disk space, it may be smarter to move some files around, or to add additional disks to get the space you need.

Friends don't let friends shrink databases (Most of the time).

Need help with this or an of the other DBCC commands? The team at Stedman Solutions, LLC specializes in repairing corrupt databases and heping when things go wrong.