



*The World's Largest Community
of SQL Server Professionals*



Top 10 Database Maintenance Best Practices

Brad M. McGehee

Director of DBA Education

Red Gate Software

www.bradmcgehee.com/presentations

My Assumptions About You

- You may be a part-time or full-time DBA.
- You may be a DBA Administrator or DBA Developer.
- You probably have less than one years experience working with SQL Server.
- If you have been a DBA for one or more years, then you are probably already familiar with much of this content. On the other hand, you might still pick up a new tip or two.

What We Are Going to Learn Today

1. Eliminating Physical File Fragmentation
2. Managing MDF Files
3. Managing LDF Files
4. Managing TEMPDB
5. Maintaining MSDB
6. Managing Indexes
7. Maintaining Statistics
8. Checking for Corruption
9. Maintaining Backups That Will Restore
10. Managing Maintenance Jobs

Not every recommendation may be appropriate for your environment. Be sure to test in a non-production environment before trying these recommendations out.

Physical File Defragmentation

- When the OS writes a file to an array, it tries to write to contiguous clusters. If contiguous clusters are not available, data is written elsewhere on the array.
- When a file is stored in a non-contiguous manner on an array, then the file is considered to be physically fragmented.
- Physical file fragmentation can contribute to an additional load on your I/O subsystem and reduce I/O performance because the disks in the array have to work harder (thrash) to read and write data.

Physical File Defragmentation

- The amount of file fragmentation's negative affect on SQL Server's performance depends on many different factors.
- For example, random reads/writes are less affected by fragmentation than sequential reads/writes.
- SANs/SSDs are often less affected by fragmentation than when using local or locally-attached storage arrays.
- Since we know fragmentation can affect SQL Server I/O performance, our goal should be to minimize it as much as possible, even though it might be hard to quantify.

Physical File Defragmentation

- SQL Server is designed to minimize physical file fragmentation, assuming the DBA is smart about the way he or she manages physical files.
- Ways to minimize physical file fragmentation:
 - Ensure there is no physical file fragmentation before creating new (or growing) database and log files.
 - Pre-size MDF and LDF files instead of letting autogrowth automatically size files. This prevents physical file fragmentation. More on this later.
 - Arrays used for backups can become fragmented over time, and you should regularly defragment them.

Managing MDF & LDF Files

- There is a myth that MDF and LDF files “manage” themselves.
- In reality, for optimal performance, DBAs must take full responsibility for managing them. This includes (more details coming up):
 - Pre-sizing MDF and LDF files appropriately
 - Monitoring MDF and LDF file growth, and manually resizing them proactively
 - Setting appropriate autogrowth settings
 - Thinking carefully before shrinking MDF and LDF files

Managing MDF Files

- When creating a new database, it is important it be pre-sized to its future expected size at least one year ahead.
- Even though only a fraction of the database will be used after the database is in production, that's not a problem.
- Your guesstimate will probably not be accurate, and that's OK. If you monitor the amount of data growth, you will soon see how accurate your guesstimate was.
- If you underestimated, you can manually grow the database to a larger size using the trending information you have collected as you have observed data growth over time.
- If you overestimated the database's size, that's not a problem either. Just let the data continue to grow inside the database, and eventually your application will use it. Don't shrink it.

Managing MDF Files

- While there are many benefits for pre-sizing databases, there are two important ones you need to be aware of.
 - First, by pre-sizing databases, SQL Server won't have to depend on autogrowth to grow the database for you, preventing the performance hit when autogrowth occurs.
 - Second, each autogrowth can contribute to physical file fragmentation, which can hurt IO performance.
- I have seen databases with 10 of thousands of autogrowths, creating serious physical file fragmentation.
- Regularly monitor database growth, and if you see that empty space is running out, then manually grow the file to its estimated size a year from now, & repeat as needed.

Shrinking MDF Files

- Don't change the default Auto Shrink database option (Set `sp_dboption` to false, which is the default).
- Don't schedule database or file shrinking operations.
- If you must shrink a MDF/NDF file:
 - Do so manually
 - Rebuild the indexes after the shrink is complete, as shrinking causes significant index fragmentation
 - Schedule these steps during the slow time of the day

Managing LDF Files

- LDF files, like MDF files, should be presized to their expected size; and at any one time, they may only be partially full, which is OK.
- Calculating the estimated log size for a newly created database is difficult to calculate, as there are so many factors that affect the size of LDF files. (e.g. T-log backup schedule, level of activity)
- I use a guesstimate of around 1-10% of the expected size of the database files one year from now. If I guesstimate that the total size of the MDF and NDF files will be 500GB one year from now, then I would estimate the LDF file to be between 5 GB and 50 GB.
- I tend to use a lower percentage for large databases and a higher percentage for smaller databases. A guesstimate is better than letting autogrowth grow the LDF file for you.

Managing LDF Files

- If you underestimate the LDF file size of a database, there are two possible ways to deal with it:
 - Regularly monitor database growth, and if you see that empty space is running out, then manually grow the file as needed.
 - If you overestimated the database's size, that's not a problem either. Unless it is a huge amount of unused space, I would generally not reduce the size of the log, leaving it at its overestimated size.
- On rare occasions, LDF files will grow wildly. If so, first determine the cause and fix it. If the grow was so wild that the LDF size is much greater than needed, and you don't expect more wild growth spurts, then you may want to consider manually shrinking it.

Managing LDF Files

- While there are many benefits for pre-sizing transaction logs, there are three important ones you need to be aware of.
 - First, by pre-sizing databases, SQL Server won't have to depend on autogrowth to grow the database for you, preventing the performance hit when autogrowth occurs.
 - Second, each autogrowth can contribute to physical file fragmentation, which can hurt IO performance.
 - Third, each autogrowth that occurs creates what are called virtual log files (VLFs). Too many VLFs in a log file can cause performance issues. By preventing autogrowths, you also prevent having too many VLFs.

Managing VLFs in LDF Files

- LDF files are not divided into pages like MDF files. Instead, they are divided into VLFs of uneven sizes from small to large. They are used to determine which parts of a log can be set to reusable (released) after a transaction log backup.
- By default, the number of VLFs created whenever the LDF is grown is:
 - Amounts less than 64 MB = 4 VLFs
 - Amounts of 64 MB to less than 1 GB = 8 VLFs
 - Amounts of 1 GB or larger = 16 VLFs
- In other words, whenever you manually grow a log file, or every time an autogrowth occurs, more VLFs are created.

Reducing VLFs in Log Files

- To determine the number of VLFs currently in a log file, run “DBCC LOGINFO (database_name)”.
- The number of rows returned is equal to the number of VLFs.
- If the number of VLFs exceed 100-200 (depending on how large the log file is), consider manually shrinking the log file, and then manually growing it to create a more “optimal” number of VLFs for your environment.
- For more information on how to do this, visit:
 - <http://sqlskills.com/blogs/kimberly/post/8-Steps-to-better-Transaction-Log-throughput.aspx>
 - <http://www.sqlskills.com/BLOGS/KIMBERLY/post/Transaction-Log-VLFs-too-many-or-too-few.aspx>

Autogrowth Settings for LDF & MDF Files

- Autogrowth should not be used to manage file growth.
- Autogrowth should only be used to cover unexpected file growth.
- Autogrowth, by default, is set to grow the MDF by 1MB at a time, and the LDF is set to grow by 10% at a time. These are poor default values.
- Instead, *set autogrowth to grow by a fixed amount you choose, and not a percentage amount.*
- Choose a fixed amount that won't result in a lot of autogrowth events, but is not so large that it will create a lot of space that won't be used in the immediate future. 1% to 10% of file size is a good starting point for a fixed amount.

TEMPDB Maintenance

- Pre-size tempdb so autogrowth doesn't have to happen often (8MB is default, which is very low).
- Set autogrowth to avoid many growth spurts, use a fixed amount that minimizes autogrowth use. (10% is default, which causes lots of autogrowth).
- If tempdb is very active, locate it on its own disks.
- If very active, consider dividing the tempdb into multiple physical files so that the number of files is $\frac{1}{4}$ to $\frac{1}{2}$ the number of CPU cores, up to 8 files. Each physical file must be the same size.

MSDB Maintenance

- Over time, the MSDB database can grow large, storing old data you probably don't need, like:
 - Backup and restore history (sp_delete_backuphistory)
 - SQL Server Agent Job history (sp_purge_jobhistory)
 - Maintenance Plan history (sp_maintplan_delete_log)
- You should periodically clear out the older data to prevent the msdb from growing unnecessarily large.
- Consider a scheduled job.

Regularly Review Indexing Needs

- Indexing Needs Change Over Time
 - Over time, data in databases, and the use of the data, often changes.
 - This means that the current indexing scheme may need to be changed over time.
 - For examples, indexes may need to be added, modified, or removed for optimal query performance.
- You need to proactively monitor your servers to see if their indexing needs are properly meet.

Identify Missing Indexes

- Other than manual index tuning, which is the ideal solution, one way to identify missing indexes is to use Profiler/SQL Trace to capture a trace file, and then use the Database Engine Tuning Advisor (DTA) to analyze the trace to look for index recommendations.
- When capturing a Profiler/SQL Trace, use the Tuning template and capture data over a representative time frame.
- Run the DTA against the trace data, review recommendations, and then add appropriate indexes.
- Note: you can use `sys.dm_db_missing_index_details` to help identify missing indexes, but it has many limitations, and I don't recommend using its recommendations unless you really know what you are doing.

Identify Unused Indexes

- Most databases have one or more indexes that were created because they seemed that they might be useful, but they have ended up not being used.
- Because indexes need to be maintained when data changes in a table, maintaining indexes that are not used is a waste of resources.
- Periodically, identify unused indexes and remove them.
- Use the `sys.dm_db_index_usage_stats` DMV to help you identify unused indexes.
- Keep in mind that the data in this DMV is cleared out each time SQL Server is restarted, so only run this DMV after the server has been up and running for quite some time.

Identify Duplicate Indexes

- For many different reasons, it is possible for the redundant indexes to be recreated using different names.
- This is resource wasteful and duplicate indexes should almost always be removed.
- See the following URL for sample scripts:

www.sqlblog.com/blogs/paul_nielsen/archive/2008/06/25/find-duplicate-indexes.aspx

<http://blogs.msdn.com/b/mssqlisv/archive/2007/06/29/detecting-overlapping-indexes-in-sql-server-2005.aspx>

<http://www.sqlskills.com/BLOGS/KIMBERLY/post/RemovingDuplicateIndexes.aspx>

Index Maintenance: Fragmentation

- Index Fragmentation Hurts Performance
 - Over time, as indexes are subjected to data modifications, gaps in data on pages develop, and the logical ordering of the data no longer matches the physical ordering of the data. Together, this is referred to as index fragmentation. This is a normal behavior, but must be regularly addressed.
 - Heavily fragmented indexes can lead to poor query performance, especially if scans occur regularly. This is because less data can fit into the data cache and because more disk I/O is required.
 - Because of this, it is important that DBAs regularly detect and remove index fragmentation from their databases on a regular basis.

How to View Index Fragmentation

```
SELECT    d.name,  
          s.OBJECT_ID,  
          s.index_id,  
          s.index_type_desc,  
          s.avg_fragmentation_in_percent,  
          s.avg_page_space_used_in_percent  
FROM      sys.databases AS d  
INNER JOIN sys.dm_db_index_physical_stats (NULL,  
NULL, NULL, NULL, 'SAMPLED')  
          AS s ON d.database_id = s.database_id  
WHERE     d.NAME = 'AdventureWorks'  
ORDER BY s.avg_fragmentation_in_percent DESC
```

Results of Fragmentation Levels

	name	OBJECT_ID	index_id	index_type_desc	avg_fragmentation_in_percent	avg_page_space_used_in_percent
1	AdventureWorks	62623266	2	NONCLUSTERED INDEX	66.6666666666667	83.7039782554979
2	AdventureWorks	98099390	2	NONCLUSTERED INDEX	66.6666666666667	41.0921670373116
3	AdventureWorks	901578250	2	NONCLUSTERED INDEX	66.6666666666667	66.5843093649617
4	AdventureWorks	1461580245	3	NONCLUSTERED INDEX	66.6666666666667	99.2257721769212
5	AdventureWorks	1749581271	1	CLUSTERED INDEX	66.6666666666667	73.1776624660242
6	AdventureWorks	1813581499	2	NONCLUSTERED INDEX	66.6666666666667	72.1522115147022
7	AdventureWorks	1989582126	1	CLUSTERED INDEX	66.6666666666667	73.1776624660242
8	AdventureWorks	2050106344	1	CLUSTERED INDEX	66.6666666666667	90.8162466024216
9	AdventureWorks	2105058535	2	NONCLUSTERED INDEX	66.6666666666667	96.7136150234742
10	AdventureWorks	1893581784	1	CLUSTERED INDEX	57.1428571428571	92.5871015567087
11	AdventureWorks	2133582639	1	CLUSTERED INDEX	50	87.0583148010872
12	AdventureWorks	2098106515	1	CLUSTERED INDEX	50	85.9154929577465
13	AdventureWorks	1461580245	2	NONCLUSTERED INDEX	50	85.2606869285891
14	AdventureWorks	2050106344	2	NONCLUSTERED INDEX	50	89.7084259945639
15	AdventureWorks	1461580245	4	NONCLUSTERED INDEX	50	71.5838893007166
16	AdventureWorks	1547152557	1	CLUSTERED INDEX	50	80.3434642945392
17	AdventureWorks	901578250	3	NONCLUSTERED INDEX	50	51.3219668890536
18	AdventureWorks	1109578991	1	CLUSTERED INDEX	50	73.4247590808006

Index Fragmentation Maintenance

- There are three ways to remove fragmentation from an index:
 - **Reorganize**: online (Standard and Enterprise Edition)
 - **Rebuild**: offline (Standard and Enterprise Edition)
 - **Rebuild**: online (Enterprise Edition Only, not covered in this session)
- Each option has its pros and cons. You must select the option(s) which work best for your environment.

Index Maintenance—Reorganize

- Removes much fragmentation and empty space, but not all.
- Only reduces fragmentation if necessary, unlike Rebuild.
- This is an online task that doesn't block user activity.
- Less MDF & LDF space is required to Reorganize than Rebuild.
- Can be stopped and started without losing work.
- Less resources are required to Reorganize than Rebuild, although it may take longer to complete.
- Index & column statistics are not updated (must perform this task separately).

Index Maintenance—Rebuild

- Virtually all fragmentation is removed.
- Index is rebuilt from scratch, and old index is dropped.
- Index statistics are updated with a FULLSCAN.
- Index rebuild is atomic on a table basis, and can't be stopped and restarted.
- More physical resources are required than Reorganize.
- Additional MDF and LDF space is required than Reorganize.
- Considered to be an off-line activity (unless you have EE).
- While index statistics are automatically updated, column statistics must be updated separately.

Defrag Recommendations

- Only Reorganize or Rebuild indexes that need it, don't defrag all indexes all the time.
- Some general recommendations (from BOL). Use as a starting point to determine what is best for your server.
 - If fragmentation is less <5%, then leave alone.
 - If fragmentation is >5% and <30%, consider Reorganize.
 - If fragmentation >30%, consider Rebuild.
- Use `sys.dm_db_index_physical_stats` to help you determine if an index should be rebuilt or not.
- Reorganize or Rebuild jobs should ideally be scheduled as a SQL Server Agent job using a custom T-SQL script you create to meet your environment's specific needs.
- Perform as needed to minimize negative effect of index fragmentation, but not more than required.

Sample Maintenance Scripts

- Sample scripts to identify indexes that need to be defragged, and how they are to be defragged (among other maintenance tasks):

<http://ola.hallengren.com/>

<http://sqlfool.com/2011/06/index-defrag-script-v4-1/>

http://www.grics.qc.ca/YourSqlDba/index_en.shtml

Maintaining Statistics

- Ensure that Auto Create Statistics and Auto Update Statistics are set to true for all databases.
- If Rebuilding indexes, index statistics are automatically updated with FULLSCAN, so you don't need to update index statistics separately, but you need to update columns statistics separately.
- If Reorganizing indexes, index & column statistics are not automatically updated, so you should update both of them manually afterwards.
- Index and/or column statistics can be rebuilt using UPDATE STATISTICS (ideally with FULLSCAN).

Checking for Corruption

- DBCC CHECKDB checks the logical and physical integrity of all objects in a database.
- Ideally, the command should be run before a full database backup is made to identify problems before the backup occurs.
- If a problem is detected, you want to identify, and correct it, as soon as possible.
- DBCC CHECKDB has some very limited “fixing” ability, but it should not be counted upon, and only used by experts, as bad data is dropped.
- Running DBCC CHECKDB is resource-intensive and should be run during slow times on the server.
- If you don't have a large enough window to run CHECKDB, restore database to another server and run CHECKDB there.

Data Corruption Detection

```
DBCC CHECKDB ('DATABASE NAME') WITH  
NO_INFOMSGS, ALL_ERRORMSGS
```

Note: Use `PHYSICAL_ONLY` option for large or busy production servers.

`--WITH NO_INFOMSGS` means that only errors messages will be displayed

`--ALL_ERRORMSGS` displays all errors found, otherwise only the first 200 error messages per object are displayed.

`--PHYSICAL_ONLY` tells the command to run in a lighter-weight mode, so fewer resources are consumed, but it may not find all problems.

Maintaining Backups That Will Restore

- Production databases should use the Full Recovery model.
- Create a job to perform full backups daily on all system and user production databases, plus log backups hourly (or similar schedule that best meets your HA needs).
- Always backup using RESTORE WITH VERIFYONLY to help verify backup integrity. But this is not a guarantee the backup is good.
- Randomly restore backups to verify that you can restore your databases. Perform daily.
- Create, and enforce, an appropriate data retention policy.
- Store backups securely (physically & encrypted), and off-site.
- If you have a limited backup window, or have limited disk space, use backup compression. Can be a big time and money saver.

Managing Maintenance Jobs

- As much as practical, keep maintenance plans the same from instance to instance.
- Don't duplicate maintenance tasks (e.g. Rebuild indexes, then update index statistics immediately thereafter).
- Schedule jobs so that they do not overlap one another.
- Schedule database maintenance tasks during down times or during the least busy time of the day.
- Don't over-maintain your databases. Find the right balance.

Take Aways From This Session

- Implementing optimal maintenance plans can greatly affect a SQL Server instances:
 - Availability
 - Performance
- Database maintenance is an on-going task that never ends. Automate as much as possible to free up your time for more interesting tasks.
- A Challenge to You: When you get back to work, evaluate all of your SQL Server instances to ensure that all appropriate maintenance tasks are being performed, and are being performed optimally.

Find Out More

Free E-Books:

- www.sqlservercentral.com/Books

Check these out:

- www.SQLServerCentral.com
- www.Simple-Talk.com

Contact me at:

bradmcgehee@hotmail.com

Blogs:

www.bradmcgehee.com

www.twitter.com/bradmcgehee



[Click Here for a free 14-day trial of the Red Gate SQL Server Toolbelt](#)



*The World's Largest Community
of SQL Server Professionals*

Thanks for Attending

Visit www.sqlservercentral.com for free SQL Server eBooks, articles, videos, blogs, news, and more.

Please Don't Forget to Turn in Your Evaluations

