



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

Database Maintenance Essentials

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

- What is Database Maintenance?
- Why is Database Maintenance Important?
- How Does the Maintenance Plan Wizard Fit In?
- Key Database Maintenance Tasks (our focus)
- Database Maintenance Best Practices

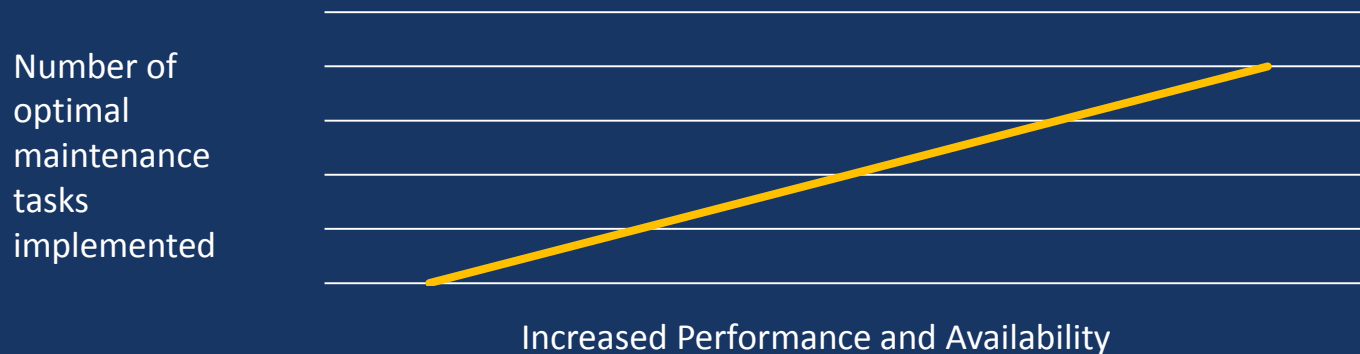
What we are going to cover today includes basic, routine database maintenance tasks that tend to be boring to perform, but nevertheless, are important tasks that need to be done. The focus of this session is on **what to do, not how to do it.**

What is Database Maintenance?

- Database maintenance includes a defined set of proactive tasks that DBAs need to perform on a periodic basis to help ensure that their databases:
 - Perform Optimally
 - Maintain High Availability
- In other words, if you don't perform proper database maintenance, performance and high availability will suffer.

Everything Counts

- While many of the maintenance tasks discussed today might seem small, the **accumulative effect** of following each recommendation can be huge.
- By following these best practices, SQL Server **performance and availability** can be boosted significantly.



How Does the Maintenance Plan Wizard Fit In?

- If used properly, the Maintenance Plan Wizard/Designer can be used to create **basic**, although **incomplete**, database maintenance plans.
- While it is designed with the “novice” in mind, unfortunately:
 - It is harder to use than it appears to be
 - It is not very flexible
 - It can’t perform all necessary maintenance tasks
 - It allows DBAs to create maintenance plans that can:
 - Significantly hurt the performance of SQL Server
 - Give a false sense of security in regard to availability
- If you decide to use the Maintenance Plan Wizard/Designer, be sure to read my free eBook, *Brad’s Sure Guide to SQL Server Maintenance Plans*.

Key Database Maintenance Tasks

- Physical File Fragmentation
- Database and Log File Management
- msdb Maintenance
- Index Maintenance
- Statistics Maintenance
- Data Corruption Detection
- Database and Log File Protection

This list is **not designed to be all encompassing**, but does include key tasks. **Some tasks are done once, not on a periodic basis.** In other words, you need the right foundation in order to get off to a good start before you begin many of the periodic maintenance tasks above. Not all recommendations apply to every SQL Server environment.

Physical File Defragmentation

- When the OS writes a file to disk, if contiguous clusters are not available, they are written elsewhere on the disk.
- When a file is stored in a non-contiguous manner on disk, 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 disk has 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 local storage.
- 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. More on this later.
 - Disks used for backups can become fragmented over time, and you should regularly defragment them.

Physical File Defragmentation

--Determine current fragmentation in Windows Server 2008:

```
C:\>defrag c: -a -v
```

--If you need to defrag, then run:

```
C:\>defrag c: -w
```

--Defrag can be used to defrag any files that are not open.

--Need 15+% free space to work properly.

--In Windows 2008, a defrag may automatically be scheduled for you.

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\BradM>defrag c: -a -v
Windows Disk Defragmenter
Copyright (c) 2006 Microsoft Corp.

Analysis report for volume C: OS

Volume size = 174 GB
Cluster size = 4 KB
Used space = 128 GB
Free space = 45.39 GB
Percent free space = 26 %

File fragmentation
Percent file fragmentation = 2 %
Total movable files = 170,121
Average file size = 795 KB
Total fragmented files = 2,692
Total excess fragments = 17,459
Average fragments per file = 1.12
Total unmovable files = 98

Free space fragmentation
Free space = 45.39 GB
Total free space extent = 5,500
Average free space per extent = 8 MB
Largest free space extent = 12.19 GB

Folder fragmentation
Total folders = 25,320
Fragmented folders = 78
Excess folder fragments = 281

Master File Table (MFT) fragmentation
Total MFT size = 182 MB
MFT record count = 170,898
Percent MFT in use = 91
Total MFT fragments = 3

Note: On NTFS volumes, file fragments larger than 64MB are not included in the fragmentation statistics.

You do not need to defragment this volume.

C:\Users\BradM>_
```

Database and Log File Management

- In theory, MDF and LDF files “manage” themselves.
- In reality, for optimal performance, **DBAs must take full responsibility** for managing them. This includes:
 - Pre-sizing MDF and LDF files appropriately
 - Setting appropriate autogrowth settings
 - Thinking carefully before shrinking MDF and LDF files
 - Monitoring MDF and LDF file growth, and manually resizing them proactively

Ensuring Instant File Initialization is On

- Enable instant file initialization, which prevents MDF files from being zeroed out when they are grown, which allows MDF files to be created quickly. LDF files are not affected.
- Speeds up CREATE DATABASE, ALTER DATABASE, RESTORE DATABASE, Autogrowth.
- Requires SQL Server 2005/2008, and Windows Server 2003/2008 (or higher version).
- Instant file initialization is turned on if the SQL Server (MSSQLSERVER) service account has been granted the SE_MANAGE_VOLUME_NAME permission by **adding the account to the Perform Volume Maintenance Tasks security policy**. Members of the local Windows Administrator group automatically have this right.
- www.bradmcgehee.com/2010/07/instant-file-initialization-speeds-sql-server/

Locating MDF and LDF Files

- The OS, pagefile.sys, and SQL Server binaries should be located on their own disk volume.
- MDF files should be located on their own disks (LUN).
- LDF files should be located on their own disks (LUN).
- Benefits:
 - Reduces I/O contention and offers better I/O performance
 - Reduces physical file fragmentation

Pre-sizing MDF and LDF Files

- The DBA needs to **estimate the future potential size of the MDF file**, and then create the MDF file to that size.
- The DBA needs to **estimate the future potential size of the LDF file**, and then create the LDF file to that size.
- Sometimes hard to estimate. May have to make educated guess, then resize later once you have more data to work with.
- Monitor, and as needed, MDF and LDF files should be grown manually (not using autogrowth).
- Benefits:
 - Reduces physical file fragmentation.
 - Reduces unexpected autogrowth events, which can cause performance problems.
 - For log files, prevents LDF file from becoming divided into excessive multiple Virtual Log Files, which can hurt performance.

Appropriate Autogrowth Settings

- 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 MDB 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.

Shrinking MDF and LDF 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 database:
 - Do so manually
 - Rebuild the indexes after the shrink is complete
 - Schedule these steps during the slow time of the day
- Benefits of not automatically shrinking files:
 - Eliminates grow and shrink syndrome
 - Reduces physical file fragmentation
 - Reduces resources used for these operations, allowing more important tasks to use them

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.

Index Maintenance Intro

- Identify and Add Missing Indexes
- Identify and Remove Unused Indexes
- Identify and Remove Duplicate Indexes
- Identify and Eliminate (or reduce) Index Fragmentation

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 tuning**, 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 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.

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/A-new-and-improved-sp_helpindex-\(jokingly-sp_helpindex8\).aspx](http://www.sqlskills.com/BLOGS/KIMBERLY/post/A-new-and-improved-sp_helpindex-(jokingly-sp_helpindex8).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 you.

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 (unless perhaps a database is small).
- 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/2010/04/index-defrag-script-v4-0/>

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

Statistics Maintenance: The Basics

- SQL Server maintains statistics on indexes (and some non-indexed columns) which are used by the query optimizer to help produce an optimal query plan.
- If these statistics are out of date, or not fully representative of the data in a table, then the query optimizer may produce a poorly performing query plan.
- As DBAs, we need to ensure that appropriate statistics are created and stay updated.

Statistics Maintenance

- 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).

Data Corruption Detection: Basics

- There are two major causes of data corruption:
 - **Physical:** Data has been altered in a negative way, most often caused by hardware or hardware drivers.
 - **Logical:** A data relationship has been broken
- There are two ways to help identify data corruption:
 - Ensure that “checksum” is turned on for your databases.
 - Run DBCC CHECKDB on your databases as often as practical.

Data Corruption Detection: Checksum

- This database setting calculates a checksum over the contents of a page and stores the value in the page header when the page is written to disk.
- When the page is read from disk, the checksum is recomputed and compared to the checksum value stored in the page header.
- If the values do not match, error message 824 is reported to both the SQL Server error log and the Windows event log.
- When you see this error, you need to take action now.
- Offers better protection than torn page detection.

Data Corruption Detection: DBCC CHECKDB

- 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.
- 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.

Database and Log File Backup Protection

- 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.
- **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.

Maintenance Monitoring: What

- Monitor Free Space—Should have 20% or more free space
- Monitor SQL Server Logs and OS Event Logs
- Monitor Jobs
- Monitor Alerts
- Monitor Blocking and Deadlocks
- Monitor Performance

- **Key Point:** Even though it is a boring task, you need to regularly monitor your SQL Server instances for all of the above.

Maintenance Monitoring: How

- Manual checks
- SQL Server Alerts
- OS Event Log Alerts
- Performance Monitor Alerts
- Profiler/SQL Trace
- SSMS Standard Reports
- Performance Dashboard (2005)
- Data Collector (2008)
- Write your own monitoring system
- Use a third-party tool

Maintenance Best Practices

- As much as practical, keep maintenance plans the **same** from instance to instance.
- Don't **duplicate** maintenance tasks (e.g. Rebuild indexes, then Update 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