



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



#92 | OREGON 2011

Best Practices Every DBA Must Know

Brad M. McGehee

Director of DBA Education

[Red Gate Software](#)

www.bradmcgehee.com

My Assumptions About You

- You may be a DBA Administrator or DBA Developer.
- You may be a part-time or full-time DBA.
- You probably have less than one year's experience as a SQL Server DBA, but you are familiar with SQL Server basics.
- If you are an experienced DBA, then you probably are already familiar with most of this content. On the other hand, maybe you may pick up a tip or two, or may be reminded about something you need to do, but have forgotten about.

What We are Going to Learn Today

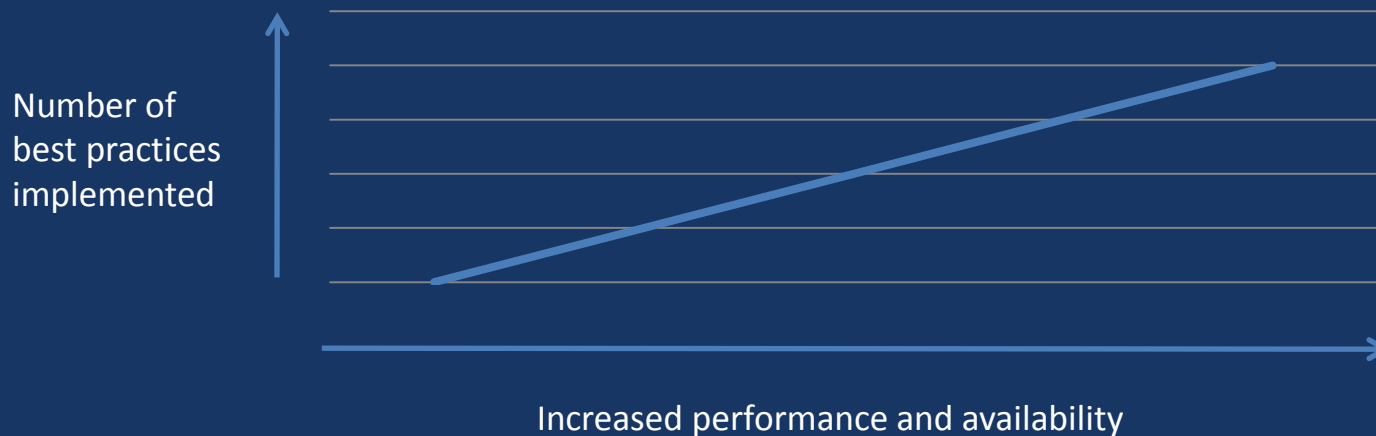
- A brief explanation why following best practices are beneficial to you as a DBA.
- Common best practices all DBAs should follow.
 - Our focus today is on what to do, not how to do it.
 - These best practices are just the basics, there is a lot more to do and learn.
 - These best practices are based on many of the common mistakes I see novice DBAs make.
 - There are always exceptions to every best practice, and not every recommendation discussed here may fit your environment. Think before you act.

Benefits of Focusing on Best Practices

- By focusing on SQL Server best practices basics, it helps you as a DBA to:
 - Optimize SQL Server performance
 - Maximize SQL Server availability

Everything Counts

- While many of the best practices I discuss today might seem small in scope, the **accumulative effect** of following each and every recommendation can be huge.
- By following best practices consistently, SQL Server **performance and availability** can be boosted substantially.



Best Practices You Should be Following

- Maintain a Standard Environment
- Installing & Upgrading SQL Server
- General Server Configuration
- Security Basics
- SQL Server Property Settings
- Memory Configuration
- User Data and Log File Management
- Don't Shrink Files
- Tempdb Management
- Database Property Settings
- Configuring Jobs—General Guidelines
- Set Up Alerts for Critical Errors
- Implement a Backup/Restore Strategy
- Create a Disaster Recovery Plan
- Document Everything
- Test Everything

Maintain a Standard Environment

- Ideally, your SQL Server environment should be as standardized as much as possible.
 - Use identical servers and hardware configurations
 - Use identical OS versions and configurations
 - Use identical SQL Server versions and configurations
 - Use identical database maintenance jobs
 - Develop an internal “Standards Guide”
- **Not always possible or appropriate**, but the more standardized your environment, the less headaches you will experience.

Installing & Upgrading SQL Server

- Generally, when **installing** a new SQL Server instance:
 - Use the newest hardware firmware and OS drivers.
 - Use the newest OS version with latest SP and patches.
 - Use the newest SQL Server version with latest SP & Hot Fixes.
 - Test, and once stable, put into production & be wary of making changes.
- Generally, when **upgrading** an existing SQL Server instance:
 - Don't upgrade **unless you have a good reason** to upgrade. If your instance is working well, don't mess with it. Don't change for the sake of changing.
 - For example, **upgrade if** you need new features, or have problems with an old installation, or need to upgrade hardware.
 - It is always safer to upgrade to a new server with a fresh installation of the OS and SQL Server than to upgrade in place. This allows you to test more effectively, and also gives you an easy **"back out" option**.

Security Basics

- Don't give users **more permissions than they need** to perform their job. (Critical. Sounds simple, often hard.)
- **Don't use the SA account** for anything. Assign it a complex password, and keep it handy just in case. Instead, use a domain account that is a member of the sysadmin role.
- Don't allow an application to use the **SA or a sysadmin** account to access SQL Server.
- Use **Windows Authentication** security whenever possible. (Especially applicable for in-house development).
- Don't give **vendors** sysadmin access to your servers.
- **Log off or lock** your SQL Server (or workstation) when done.

General Server Configuration

- Ideally, SQL Server instances should run on a **stand-alone** server (physical or virtual) with no other major apps running on it. Small monitoring apps or utilities are generally OK.
- **Avoid multiple instances** unless you have a really good reason to use them. Consider virtualization instead.
- **Unnecessary SQL Server services** should be uninstalled or turned off.
- Ideally, **don't run antivirus/antispysware** software locally.
 - If your organization's policy requires running antivirus/antispysware software locally, exclude MDF, NDF, LDF, BAK, TRN, Filestream, and BCP files.

SQL Server Property Settings

- SQL Server includes many **instance-wide** property settings. (e.g. sp_configure)
- **Don't change** any of these defaults unless you know exactly what the effect of the change is.
- There are a **few** of these settings that are commonly changed from their defaults, and generally you need to perform baselines before and after the change to see if performance and HA increased or decreased because of the change.
- If you make a change, only make one change at a time so that you can see its effect independent of other variables.

Memory Configuration

- Ideally, use the **64-bit** version of the OS and SQL Server.
- If using the **32-bit** version of SQL Server, and if using 4 GB or more of RAM, ensure that /3GB switch and AWE memory are correctly configured. Correct settings depend on available RAM and other factors.

Data and Log File Management

- Remove **physical file fragmentation** before creating new or expanding existing MDF or LDF files.
- When creating new MDFs and LDFs, **pre-size** them to eliminate/minimize autogrowth events.
- **MDF** files should be located on their own disks.
- **LDF** files should be located on their own disks.
- BAK and TRN **backup files** should be located on their own disks.

Instant File Initialization

- **Enable instant file initialization**, which prevents MDF files from being zeroed out when they are grown, 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 service account to the Perform Volume Maintenance Tasks security policy**. Members of the local Windows Administrator group automatically have this right.

Don't Shrink Files

- If you properly size your MDFs and LDFs, then you should very rarely have to shrink a file.
- Don't schedule periodic database or file shrinking operations.
- If you must shrink a database (MDF file):
 - 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 Management

- **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.

Database Property Settings

- Don't change database property settings unless you have a very good reason. Some key ones:
 - **Auto Create Statistics:** On
 - **Auto Update Statistics:** On
 - **Auto Shrink:** Off
 - **Autogrowth:** Leave on. Use mainly for catching mistakes. File growth should be managed manually. Use fixed amount that minimizes autogrowth occurrences.
 - **Recovery Mode:** Set to full for all production databases so transaction log backups can be made.
 - **Page Verify:** Use Checksum (2005/2008), don't turn off.
 - **Compatibility Level:** Should be set to match current server version, unless there are compatibility problems.

Configuring Jobs—General

- If your production servers **don't have any jobs**, then there is a problem, as all servers need jobs.
- Try to schedule jobs so they **don't interfere** with production.
- Try to prevent jobs from **overlapping**.
- Set **alerts** on jobs so you are notified if they fail.
- **Check jobs** daily to verify that they have run correctly (not hung, not run abnormally long, etc).
- If you use the Maintenance Plan Wizard, **be careful** to use it properly. If misused, it can create maintenance jobs that hurt performance.

Create Index Rebuilding/Reorganize Job

- Indexes need to be **rebuilt or reorganized regularly** to minimize fragmentation and reduce wasted space.
- 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 instance.
 - 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.

Create Data Corruption Detection Job

- Ideally, run DBCC CHECKDB as **frequently as practical**.
- If you have a problem, you want to find it as soon as possible to reduce the risk of data loss.
- Create an appropriate job to run this (or similar) command:

```
DBCC CHECKDB ('DATABASE_NAME') WITH NO_INFOMSGS, ALL_ERRORMSGS;
```

Note: Consider using PHYSICAL_ONLY option for large or busy production servers to reduce run time.

- **Don't use the DBCC CHECKDB repair option** unless you fully understand its implications, as data loss will most likely occur.

Implement a Backup/Restore Strategy

- Create a job to perform **full backups daily** on all system and user production databases, plus **log backups hourly** (or similar variation). Include differential backups if appropriate.
- If a database **uses the bulk or full recovery model**, you must back up the transaction log to keep it from growing uncontrollably.
- Backup using **RESTORE WITH VERIFYONLY** to help verify backup integrity. (Does not guarantee good backups.)
- Periodically **test backups** to see if they can be restored.
- Set up an appropriate backup **retention policy**.
- Store backups **securely and off-site** (not on same disk array or SAN).
- If you have a limited backup window, or have limited disk space, use **backup compression**. Can be a big time saver.

Sample Maintenance Scripts

- Sample database maintenance scripts to check out:
 - <http://ola.hallengren.com/>
 - <http://sqlfool.com/2011/06/index-defrag-script-v4-1/>
 - http://www.grics.qc.ca/YourSqlDba/index_en.shtml

Set Up Alerts for Critical Errors

- Create a **SQL Server Event Alert** for all events with a severity of 19 [fatal] and higher.
- **Have alerts sent to you** or whoever is responsible for day-to-day monitoring.
- Consider a **third-party alerting tool** if SQL Server Alerts doesn't meet all of your needs.

Create a Disaster Recovery Plan

- **You must** create a document that outlines, step-by-step, in detail, how you will recover your SQL Servers in the case of any problem, small or large.
- You need to **practice** using the plan so you are familiar with it and can easily implement it.
- Keep **Microsoft SQL Server's Product Support** phone number handy. Paste it near your computer.
- **Remember:** Most “disasters” are small, such as a corrupted database. Big “disasters” occur very rarely, if ever. But you **need to be prepared** for both.

Document Everything

- Yes, documentation is very boring, but it is very critical to being a successful DBA. Be sure to document:
 - The installation and configuration of each instance.
 - The installation and configuration of any application that uses SQL Server as its back end (as related to SQL Server).
 - Troubleshooting tasks, as the same problem may reoccur, and you don't want to reinvent the wheel.
 - Any time any change is made to any instance for any reason.
- Be sure that documentation is easily available to everyone who needs access to it.

Test Everything

- Before you make any change on a production SQL Server, be sure you **test it first in a test environment.**
 - NO EXCEPTIONS!
 - I mean it!
 - Really!
 - No kidding.
 - I wouldn't lie to you.
 - You don't want to lose your job.
 - You'd be crazy not listening to this advice.
 - Civilization as we know it may lie in your hands.
- Never forget, DBAs are the **protectors of the organization's data.** You took this oath when you accepted the job of DBA (whether you like it or not).

Take Homes for Today

- By **focusing on best practices**, you gain the following:
 - Better SQL Server performance
 - Higher SQL Server availability
- The **total effect** of following each and every recommendation made today can be huge.
- What you learned today is only the **tip of the iceberg**, you will need to take time to learn many other best practices.
- **Challenge to you:** when you get back to work, use this as a mini **checklist** to give your SQL Servers a quick health check.

Find Out More

- Free E-Books on SQL Server:
 - www.sqlservercentral.com/Books
- Check these websites out:
 - www.SQLServerCentral.com
 - www.Simple-Talk.com
- Contact me at:
 - bradmcgehee@hotmail.com
- Blogs:
 - www.bradmcgehee.com
 - www.twitter.com/bradmcgehee

