**Database Administration Considerations**

Written by Ryan Redburn

**Preamble**

This document is not meant to be a comprehensive guide to database administration, but rather a list of items I think are relevant to administration and deserving of attention. Some of the external materials linked in this document do provide recommendation and guidelines, but should not necessarily be considered all-encompassing. If you require more detailed material there are numerous sources online and off that will give you what you seek.

**Main Contents**

**Server Workload**

Knowing what the server workload looks like will help you make decisions regarding various activities (e.g., what maintenance tasks should be performed, how closely should performance be monitored and issues remedied, etc.). You want to be aware not just of want workload patterns and resource consumption by SQL Server, but also any other applications that may be running on a server.

**Server Versions/Editions**

Ideally all servers will be on a fairly recent service pack/cumulative update and running a supported [version](https://support.microsoft.com/en-us/help/321185/how-to-determine-the-version-edition-and-update-level-of-sql-server-an) of SQL Server. Additionally, you will want to know what editions are in use so you can assess what features are available and what limitations exists. Depending on the point in time you are reading this, it may also be important to consider the various engine changes made my Microsoft starting with SQL Server 2014.

Database Engine Changes

* 2014: [Cardinality Estimation Changes](https://docs.microsoft.com/en-us/sql/relational-databases/performance/cardinality-estimation-sql-server?view=sql-server-2017)
* 2016: [Auto Update Statistics Threshold Change](https://docs.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-2017)
* 2017: [Adaptive Query Processing](https://docs.microsoft.com/en-us/sql/relational-databases/performance/adaptive-query-processing?view=sql-server-2017)

Edition Feature Comparisons

* [2008 R2](https://msdn.microsoft.com/it-it/library/cc645993(v=sql.105).aspx)
* 2012 (unavailable)
* [2014](https://docs.microsoft.com/en-us/sql/getting-started/features-supported-by-the-editions-of-sql-server-2014?view=sql-server-2014)
* [2016](https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2016?view=sql-server-2017) (Significant differences to previous versions starting at SP1)
* [2017](https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2017?view=sql-server-2017)

**Product SLA(s)**

Here we’re thinking about things such as, but not limited to, recovery objectives, data retention, and access control. A product SLA will likely influence many operational areas and everyone working on the product should be aware of the relevant terms.

**Existing Maintenance Operations**

Particularly when planning and implementing new maintenance operations, but even just more generally, you should be aware of what maintenance tasks are already being performed. This may elucidate opportunities to consolidate maintenance efforts, remove useless tasks, implement missing tasks, and plan/evaluate maintenance windows.

**Server Settings**

Generally there are only a handful of server settings that should be changed as part of regular server setup. Unless you have a specific reason (and evidence to back it up) you should not be changing the more esoteric settings (e.g., Optimize for Ad hoc Workloads). These blog posts from Brent Ozar Unlimited on [memory settings](https://www.brentozar.com/blitz/max-memory/) and other [miscellaneous settings](https://www.brentozar.com/archive/2013/09/five-sql-server-settings-to-change/) are good starting points.

**Database Settings**

Similar to server settings, you should not be changing database settings unless you have a really good reason. The one obvious exception is the recovery model; you should change this as appropriate to match your backup/recovery strategy.

Other settings that should often be changed on the growth settings on all the database files. These should be set to reasonable values. See [this](https://www.brentozar.com/blitz/blitz-result-percent-growth-use/) blog post from Brent Ozar Unlimited.

There are a couple settings to watch when upgrading to earlier versions, as you may need to change them depending on your particular migration path. Page Verify should be set to CHECKSUM and the compatibility level should match the instance version unless you have a reason to keep it back. Make sure to test any relevant applications before moving the compatibility level forward.

**Tempdb Configuration**

There are a few changes you typically want to make to tempdb for optimal performance. These include things such as having multiple, evenly sized data files and moving tempdb onto a separate drive. See [this](https://www.brentozar.com/archive/2014/06/trace-flags-1117-1118-tempdb-configuration/) blog post from Brent Ozar Unlimited and [this](https://blogs.msdn.microsoft.com/sql_server_team/tempdb-files-and-trace-flags-and-updates-oh-my/) one from the SQL Server engineering team. Note that numerous changes have been made to the default tempdb configuration by Microsoft over the years (the previously linked Microsoft blog details some of these). If you’re looking for guides and/or information on configuring tempdb make sure you look at recent sources when possible.

**Table Types**

In an OLTP system every table should be clustered unless you have a specific, quantifiable reason to use a heap. In an OLAP system clustered [columnstore](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?view=sql-server-2017) is also an option.

**Data Model and Distribution**

It is good to have at least a general idea of what the data model and distribution looks like for any given database. This will help with maintenance activities such as data pruning, archiving, and performance tuning.

**Disabled Indexes/Constraints**

Constraints and indexes exist for a reason and you should make sure that none are disabled. There are some valid scenarios (e.g., ETL processes) where these objects may be disabled, but it would typically only be for a set period of time. Be careful enabling indexes/constraints that you find disabled, as that may have undesirable consequences on the system.

**Index Usage and Missing Indexes**

Index usage and missing indexes are pieces of information you should periodically examine. Adding and removing indexes is one of many things that can be done to improve the health of the system. Make sure when doing index analysis to have data from a period of time that accurately captures all the various workloads handled by the server. Having a comprehensive dataset helps avoid removing indexes which may be needed by infrequently run processes and adding indexes which are less valuable than a few days’ worth of data may tell you.

**Table/Index Fragmentation**

Monitoring and potentially addressing fragmentation isn’t necessarily a fruitless endeavor, but in the modern era this is less useful than in times past. Often times a significant portion, if not the entirety, of data can be kept in memory. [This](https://groupby.org/conference-session-abstracts/why-defragmenting-your-indexes-isnt-helping/) video from Brent Ozar details why defragmenting may not be helping (and may even be detrimental).

**Server Wait Stats**

It’s a good idea to monitor instance wait stats. This may help you identify recurring problems (e.g., queries going parallel when they shouldn’t) and/or identify potential problems that haven’t surfaced in an obvious manner. [This](https://www.brentozar.com/sql/wait-stats/) blog post from Brent Ozar Unlimited and [this](https://www.sqlskills.com/blogs/paul/wait-statistics-or-please-tell-me-where-it-hurts/) one from Paul Randel are good starting points for analyzing wait stats.

**Expensive Queries, Locking, and Blocking**

Monitoring expensive queries, locking, and blocking will give you an idea of where to best direct performance tuning efforts. Similar to index tuning, this should be done periodically to ensure optimal system performance.