SIEMENS
Ingenuity for life

Microsoft SQL Server for Teamcenter Best Practices

Version 2.5

Advanced Technical Services White Paper

This whitepaper describes Teamcenter's best practices for use with a Microsoft SQL Server Database.

www.sw.siemens.com

© Siemens 2022

Disclaimer

This document is intended to provide guidance and best practices for Microsoft SQL Server. Siemens Digital Industries Software is providing this information as is, without warranty of any kind. **SIEMENS DIGITAL INDUSTRIES SOFTWARE hereby disclaims and assumes no responsibility or liability for any results that occur due to the use of the information contained in this document**.

© Siemens 2022

DOCUMENT HISTORY

Version	Date	Author	Description
1.0	11/12/2018	P. Kessler	DRAFT
1.1	3/9/2018	P. Kessler	Revision
1.2	6/5/2018	P. Kessler	Edits
1.3	6/22/2018	T. Goerdt	Edits
1.4	6/21/19	P. Kessler	Additional Content
2.0	6/2/2020	D. Howe	Additional Content
2.1	4/3/2021	D. Howe	Minor updates
2.2	26/05/2021	D. Howe	Update Max DoP advice
2.3	07/01/2022	D. Howe	Updates for SQL 2019 and TC 12+
2.4	19/08/2022	D. Howe	Minor clarifications
2.5	01/09/2022	D. Howe	Updated Cloud deployment and HA details. Added TC 14 support plus TF updates. Updated tempdb advice

Contributors

Version	Date	Contributor	Description
1.0	12/19/2018	D. Howe	Additional Content
1.1	3/9/2018	D. Howe	Additional Content
1.3	6/22/2018	T. Goerdt, E Danielson	Additional Content

Reviewed By

Version	Date	Reviewed By	Description
1.0	3/6/18	T. Goerdt	Completed Review
1.0	3/6/18	E. Danielson	Completed Review
1.1	3/9/18	D. Howe	Completed Review
1.3	6/22/2018	T. Goerdt / E. Danielson	Completed Review

© Copyright 2022

Table of Contents

© Copyright 2022

© Copyright 2022

© Copyright 2022

1 Introduction

Among the strengths of SQL Server as a database platform is the relative simplicity of managing and tuning the database engine. There are, nevertheless, a range of best practices for managing SQL Server production deployment and for getting the best performance and reliability from the platform. Many of these are standard, irrespective of the Teamcenter application, while some specific guidance is driven by the nature of the Teamcenter database and application.

The following sections discuss a range of best practices and recommended choices for Teamcenter deployments on SQL Server.

2 Planning and Installing the SQL Server Instance

Use a Dedicated Database Server and SQL Instance
Because Teamcenter is a database-intensive application, Siemens recommends hosting the SQL Server database on a server dedicated to that purpose. This will prevent the Teamcenter database workload from having to compete for CPU, memory, network, and I/O with other processes.

2.1 *Select the Appropriate SQL Server Edition*

Teamcenter 9.1 can run on either the Standard or Enterprise Edition of SQL Server 2008 R2 and TC 9.1.1 added support for SQL Server 2012.

Teamcenter 10.1 can run on either the Standard or Enterprise Edition of SQL Server 2008 R2, SQL Server 2008 R2 SP2, SQL Server 2012 and TC 10.1.5 added support for SQL Server 2014.

Teamcenter 11.1 can run on either the Standard or Enterprise Edition of SQL Server 2012, SQL Server 2014 and TC 11.3 added support for SQL Server 2016.

Teamcenter 12.x is supported with SQL Server 2014 and 2016 (Standard and Enterprise) and at TC 12.2 MS SQL Server 2017 and Server 2017 running in 2016 compatibility mode (on Windows only) was certified. AT 12.2 additional support was added for Always On failover.

Teamcenter 13.1 supported with SQL Server 2017 running in 2016 compatibility mode (on Windows only) and using legacy cardinality estimation was certified. At 13.2 SQL Server 2019 on both Windows and Linux is supported. The standard release of 2019 does not include any fixes for CE issues and legacy cardinality estimation may be required for some customers. Cumulative Patch 4 for 2019 and 19 for 2017 addresses some known CE issues. see KB4538497 - FIX: Slow query performance when using query predicates with UPPER, LOWER or RTRIM with default CE in SQL Server 2017 and 2019 (microsoft.com)

Teamcenter 14.x is supported with SQL Server 2017 and 2019 (Standard and Enterprise) on Windows and SQL Server 2019 on Linux. CU 4 is recommended.

Regardless of version and edition, you should always run on the latest SQL Server service pack, which you can find in the Downloads section of the SQL Server Developer Center. For most functional aspects of Teamcenter, SQL Server Standard and Enterprise editions are equivalent. However, the Enterprise Edition of SQL Server includes a variety of higher-end features. Consult the Microsoft website for a current list of differences for your proposed version of SQL Server as these features may change with newer releases.

2.2 *Options for High Availability*

SQL Server has several approaches for maintaining database availability in the event of a system failure. The most sophisticated methods, and the only ones that provide automatic switchover to a standby server in the case of failure, are failover clustering and database mirroring. Other options, which require more manual intervention by administrators, include log shipping and SQL Server replication.

Siemens recommends a high availability strategy based on failover clustering as the preferred approach. However, Microsoft are rapidly replacing clustering with Always On availability groups using synchronous replication and automatic failover. This method should avoid many of the failover caused by the standard clustering.

The use of database mirroring (without automatic failover) and log shipping are viable options as well especially for disaster recovery. However, administrators must be prepared for the need to manually intervene to redirect Teamcenter to a standby server if the primary server experiences a failure.

2.2.1 Always On Failover Cluster Instance

SQL Server Always On offering, “Always On Failover Cluster” Instances leverages Windows Server Failover Clustering (WSFC) functionality to provide local high availability through redundancy at the server-instance level-a *failover cluster instance* (FCI). An FCI is a single instance of SQL Server that is installed across Windows Server Failover Clustering (WSFC) nodes and, possibly, across multiple subnets. On the network, an FCI appears to be an instance of SQL Server running on a single computer, but the FCI provides the Virtual Network Names (VNN) for the cluster and handles failover from one WSFC node to another if the current node becomes unavailable.

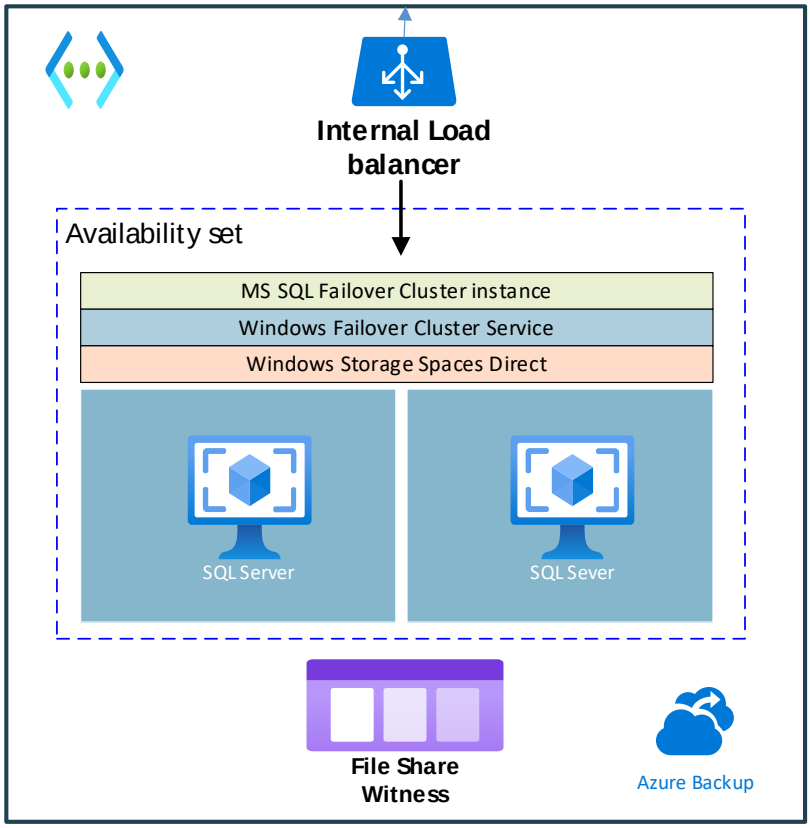
On premise the WSFC handles the failover using the standard network address resolution protocol GARP. On public cloud GARP is not supported and another method must be used. Two method are available, a load balancer and later version Distributed Network Names (DNN).

You should note, in an availability group, automatic failover from an FCI to other nodes within the availability group is not supported. This means that FCIs and standalone nodes should not be coupled together within an availability group if automatic failover is an important component your high availability solution. However, this coupling can be made for your disaster recovery solution.

2.2.1.1.1 FCI using Load Balancer.

One solution to resolve this loss of functionality is to use a load balancer to manage the failover and provide a virtual network name (VNN) for the cluster.

The diagram below shows an Azure deployment of a SQL Server failover cluster configuration using an Azure NLB to provide failover. The diagram below shows an Azure deployment of a SQL Server failover cluster configuration using an Azure NLB to provide failover.



Details of the Azure NLB configuration can be found on the Microsoft Web site. See [Configure Azure Load Balancer a failover cluster instance VNN - SQL Server on Azure VMs | Microsoft Docs]

2.2.1.1.2 FCI using DNN (Distributed Network Names)

To remove the load balancer requirement DNN was added to Window Server 2016 onwards and is supported by SQL Server 2016 SP3, 2017 CU25 and 2019 CU8.

When a DNN resource is created, the cluster binds the DNS name with the IP addresses of all the nodes in the cluster. The client will try to connect to each IP address in this list to find which resource to connect to.

A distributed network name is recommended over a load balancer when possible because:

- The end-to-end solution is more robust since you no longer have to maintain the load balancer resource.
- Eliminating the load balancer probes minimizes failover duration.
- The DNN simplifies provisioning and management of the failover cluster instance or availability group listener with SQL Server on Azure VMs

2.2.2 Always On Availability Groups

The feature known as Always On is a high availability and disaster recover feature that was fully introduced in SQL server 2012. This feature maximizes the availability of a set of user databases in an enterprise. The availability group supports a failover environment for a set of databases that fail over together. The availability group supports a set of read-write primary databases and one to four sets of corresponding secondary databases. Optionally secondary databases can be made available for read-only access and some backup operations.

There are some specific issues around using Always On and Teamcenter using asynchronous commit and automatic failover that have been addressed in Teamcenter 12.2 onwards. If a customer chooses to use the Always On feature for high availability and automatic failover it should be configured to utilize a synchronous commit. For a remote Disaster recovery system, the system should be configured with asynchronous commit for best performance.

2.3 *Which Features to Install*

SQL Server includes a rich array of components for data management and business intelligence. A minimal installation for Teamcenter requires only the SQL Server Database Engine and Management Tools. These selections are made in the Feature Selection step of the SQL Server setup. Consult the latest version of the Teamcenter Windows Server Installation documentation section on Microsoft SQL Server installation for a list of required features for your Teamcenter version.

2.4 *Proper Placement of the tempdb Database*

Tempdb is a built-in database used by SQL Server as a workspace for temporary objects. These objects could be temporary tables resulting from application code, internal sort tables used by SQL Server, or a variety of other things. Depending on the load and usage patterns in each environment, the configuration of tempdb can become a factor in overall performance. For this reason, there are some important best practices for managing this database.

These tempdb best practices can be boiled down to a few key points:

1. Move tempdb from its default location to one or more high-performance, fault-tolerant disk volumes.

2. If possible and practical given your disk resources, give tempdb files dedicated use of their disk(s).

3. To minimize auto growth, configure the startup size of tempdb based on your server's needs. Frequent auto growth can hurt performance.

4. Create multiple data files for tempdb to reduce contention in the engine's internal allocation routines.

5. Enable instant file initialization.

6. Enable any Trace Flags for example TF 3427

We'll review the guidelines around each of these and then provide an example that brings it all together.

2.5 *Moving tempdb*

By default, tempdb files are in the same directory as SQL Server's other system databases. For busy servers, this is usually sub-optimal. Tempdb should be moved to a high-performance, fault-tolerant disk volume, for example RAID 1 or RAID 10 volume on fast disks, tempdb should never be placed on a RAID 5 volume because of the inferior write performance of RAID 5.

SQL Server must be restarted for this change to take effect. Note that SQL Server creates fresh copies of tempdb data and log files every time the service starts, so there is no need to move any files yourself. Just restart the service, and you will see the files in the new location(s) you have specified.

2.6 *Using Multiple, Dedicated Disks*

If your database server has sufficient disk resources and your server is heavily utilized, typical best practices for database file placement should be followed for tempdb. The following table summarizes these best practices.

If You Have…	Then…
One disk volume available for tempdb	Place all tempdb data and log files here, preferably with no other heavily utilized files on this volume.
Two disk volumes available for tempdb	Place the tempdb log on one volume. Place all tempdb data files on the other volume.
More than two disk volumes available for tempdb	Place the tempdb log alone on one volume. Distribute tempdb data files evenly across the remaining volumes.

2.7 *Sizing tempdb*

Tempdb is used very dynamically by SQL Server. In other words, the server is constantly adding and removing structures in tempdb. At any given time, some operations may cause a (relative) spike in the required size of the database. If tempdb starts out too small given your workload, the result may be a large number of frequent auto growth operations in tempdb. This has two negative side effects. First, performance can be hurt when the auto growth occurs. Second, many small auto growths will result in heavily fragmented data files, which hurts tempdb I/O performance from that time forward.

To avoid these problems, the size of tempdb should be set proactively as appropriate for the environment. There is no fixed rule for the size of tempdb—it depends on the environment. Two potential approaches are recommended here. The first is to use an arbitrary rule of thumb, and the second is to observe your environment in action and derive appropriate sizing. DBAs should monitor usage over time to see if configuration changes become needed.

- Small Database – 2GB
- Medium Database – 5GB
- Large Database – 10GB

The second approach—observation of the environment in action—starts with leaving tempdb sizes at their default level with auto growth enabled. After a reasonable period of production activity (one or more weeks, for example), the DBA should check the space used by tempdb. The size of tempdb can be viewed in SQL Server Management Studio.

The DBA can use the size to which tempdb data and log files have grown as a good indicator of target size. Adding an additional 20% buffer can help ensure that auto growth is minimized.

Finally, whatever size you choose for tempdb, it is a good idea to leave auto growth enabled on the database. Minimizing auto growth is a goal, but you want to leave auto growth enabled so that tempdb does not run out of space in an emergency.

More discussion of sizing tempdb can be found on MSDN.

2.8 *Adding Data Files to tempdb*

By default, the tempdb database has one data file (.mdf) and one log file (.ldf). A best practice for large, multi-core servers is to increase the number of files in tempdb. This can reduce contention in SQL Server's internal allocation routines related to tempdb. There are essentially four points to consider when doing this:

1. Aim for one tempdb data file per processor core available to SQL Server. For systems with more than 16 CPUs, it is safe to relax this ratio closer to 0.5 files per core but initially no more than 8. If contention on tempdb is seen, increase the number of data files by multiples of four (4) up to the number of logical processors until the contention is reduced to acceptable levels.

2. Ensure that the data files are equally sized. This allows SQL Server's proportional fill algorithms to run at their most efficient.

3. The data files need not all be on separate disk volumes. The purpose of this recommendation is not to split I/O throughput across devices, but to exploit some aspects of the Database Engine's internal I/O and allocation algorithms. This is a bit counterintuitive and differs from the guidance for user databases, but it is intentional.

4. There is no need for multiple log files.

2.9 *Instant File Initialization*

Enable instant file initialization for the benefit of data file space allocations. This will speed up space allocations in tempdb data files, increasing query performance.

2.10 *Memory-optimized tempdb metadata*

SQL Server 2019 (15.x) introduced memory-optimized tempdb metadata to reduce metadata contention in tempdb that has been a performance bottleneck.

This feature effectively removes this bottleneck and unlocks a new level of scalability for tempdb-heavy workloads. In SQL Server 2019 (15.x), the system tables involved in managing temporary table metadata can be moved into latch-free, non-durable, memory-optimized tables. (not available on Azure SQL Managed Instance).

It is enabled with this command

```
SQL> ALTER SERVER CONFIGURATION SET MEMORY_OPTIMIZED TEMPDB_METADATA = ON;
```

For more details see: [tempdb database - SQL Server | Microsoft Docs]

2.11 *Snapshot Isolation*

Once the user number reaches around 50 to 100 you will see a lot of dead locks caused by lock escalation. You must enable read committed snapshot isolation to avoid this issue.

You must set both ALLOW_SNAPSHOT_ISOLATION ON and READ_COMMITTED_SNAPSHOT ON to make this work. Allow snapshot isolation activates the mechanism for storing row versions in the temporary database, without read committed snapshot will not work. For more details see [Snapshot Isolation in SQL Server - ADO.NET | Microsoft Docs]

Once enabled a row version for each transaction is maintained in tempdb. The versions are maintained until the transaction is complete. A simple select is considered a transaction and requires a commit to compete it.

2.11.1 Ghost records

- Records that have been logically deleted but physically exist on a page.
- Simplifies key range locking and transaction rollback.
- A record is marked as deleted by setting a bit. A ghost cleanup process physically deletes the ghost records after the calling transaction commits.

2.11.2 How to prevent Ghost Records from being an issue

The problems come from long running transactions as they build a huge number of rows version in the tempdb. So, for example IDSM can make a simple select and repeats it without a commit; the result is a huge number of row versions.

In TC 11.2.2 TC_MSSQL_SELECT_AUTOCOMMIT=True was added to automatically close transactions. TC_MSSQL_SELECT_AUTOCOMMIT=True should be added to TC_profilevars to ensure it has been enabled. In practice this has been found not to add any significant overhead. Note TC_MSSQL_SELECT_AUTOCOMMIT is defaulted ON from 13.2.

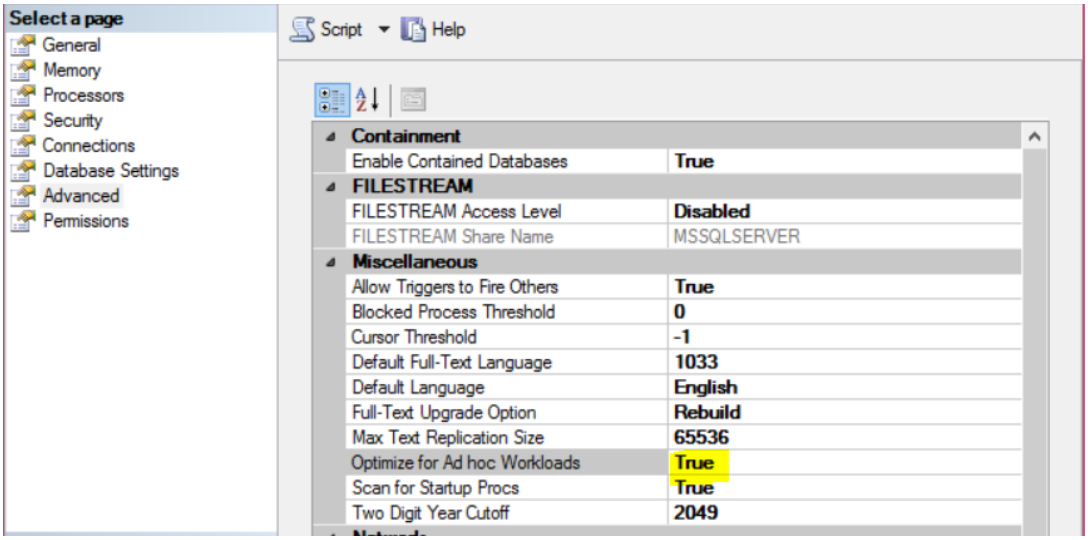
Also consider setting the index Fill factor to 20% for the POM_M_LOCK, POM_R_LOCK, PPOM_SESSION, POM_F_LOCK, POM_LOCK, POM_LOCK_LOGGING, PM_PROCESS_LIST tables. They have a high insert/delete load, which leaves a lot of ghost records when using snapshot

isolation. The Fill factor should make it easier to insert new records in the indexes without constantly tearing the pages.

2.12 *Enable Query Store*

Enable the query store with QUERY_CAPTURE_MODE set to AUTO and enable the optimize for ad hoc workloads server configuration option. The query store helps analyses performance issue and can even automatically switch back to using a prior execution plan if there's a performance problem with a new plan. Enabling the query store when you have an issue is too late, the query store collection data which it uses to amylase the performance.

Enabling the "optimize for ad hoc workload" option will control the query store size by preventing the cache of single use plans.



2.13 *Trace Flags*

- Trace Flag 1118 - Full Extents Only

  KB 2154845 advises that Trace Flag 1118 can help in some situations. That trace flag tells SQL Server that it should avoid "mixed extents" and use "full extents".

  This means that each newly allocated object in every database on the instance gets its own private 64KB of data. Tempdb is usually the place where most objects are created, so it makes the most difference there.

  Note: Starting with SQL Server 2016 (13.x) this behavior is controlled by the SET MIXED_PAGE_ALLOCATION option of ALTER DATABASE, and trace flag 1118 has no effect.

- Trace Flag 1117 – Grow All Files in Filegroup Equally

Trace flag 1117 changes the behavior of file growth: if one data file in a filegroup grows, it forces other files in that filegroup to ALSO grow. This can be useful for tempdb, which is commonly configured with multiple data files as KB 2154845 advises.

Note: Implementing this trace flag impacts every database on the instance and not just tempdb. Some administrators may prefer to pre-grow tempdb files, so they fill the tempdb drive, ensuring to leave room for any "free space monitoring" you have.

Note: Starting with SQL Server 2016 (13.x) this behavior is controlled by the AUTOGROW_SINGLE_FILE and AUTOGROW_ALL_FILES option of ALTER DATABASE, and trace flag 1117 has no effect

- Trace Flag 610 – Controls minimally logged inserts into indexed tables
  Trace flag 610 reduces the logging for bulk INSERTs into tables with indexes. Rows inserted into existing pages (no new page allocation) to maintain index order are still fully logged, as are rows that are moved as a result of page splits during the load. It is also important to have ALLOW_PAGE_LOCKS turned ON for indexes (which is ON by default) for minimal logging operation to work as page locks are acquired during allocation and thereby only page or extent allocations are logged.

  Note: Starting with SQL Server 2016 (13.x) trace flag is not required starting SQL Server 2016 as minimal logging is turned on by default for indexed tables.

- Trace Flag 3427 – Improves performance of workloads utilizing many frequent, short transactions, such as those that involve tempdb. Teamcenter cane generate many frequent transactions and will benefit from this trace flag. This flag is available from Cumulative Update 2 for SQL Server 2016 SP1 to Cumulative Update 2 for SQL Server 2016 SP2.

- Trace Flag 2371 – Changes the fixed update statistics threshold to a linear update statistics threshold. SQL Server 2016 (13.x) and under the database compatibility level 130 or above, this behavior is controlled by the engine and trace flag 2371 has no effect.

2.14 *Configuring Memory for SQL Server*

SQL Server can dynamically manage memory without intervention by administrators. In many cases, this is fine. For systems that will host large databases with heavy user traffic, however, it may be necessary to set some limits on memory usage to prevent performance degradation.

A key indicator that this may be necessary is if the size of the Teamcenter database (actual data space used) is significantly greater than the amount of RAM on the database server. Over time, memory pressure may develop. SQL Server will want to continue growing its in-memory data cache to avoid reading from disk, but the operating system will push back, trying to allocate memory for itself and other processes. The result can be excessive memory paging and a negative effect on performance.

If your Teamcenter database (data + indexes) is or is expected to be much larger than the total RAM on the database server, Siemens recommends configuring the min and max server memory options in SQL Server. Use the recommendations in the following table to select values appropriate for your environment.

Physical RAM	SQL Min Server Memory (MB)	SQL Max Server Memory (MB)
8GB	4,096	5,120 (5GB)
16GB	8,192	12,288 (12GB)
24GB	12,288	18,432 (18GB)
32GB	16,384	25,600 (25GB)
48GB	32,768	39,936 (39GB)
64GB	49,152	56,320 (55GB)
96GB	73,728	88,064 (86GB)
128GB	104,448	120,832 (118GB)
256GB	235,520	251,904 (246GB)

The min/max server memory options can be set in the Server Properties dialog box in SQL Server Management.

2.14.1 Lock pages (VMs and Cloud)

Another option to consider is granting the **Lock pages in memory** permission to the SQL Server service account. This permission allows SQL Server to lock its buffer memory and refuse to page it out in response to requests from the operating system. This can give the Database Engine additional ability to preserve its caches and maintain performance, especially if other processes are competing for resources on the server.

Lock pages should be considered when SQL Server is deployed on a virtualized server or on the Cloud (Azure or AWS). Lock pages prevents the hyper-visor reducing the memory.

For a physical host Siemens recommends hosting the Teamcenter database on a server dedicated to this purpose. If this recommendation is followed, the importance of the **Lock pages in memory** permission is lessened. In addition, improvements in memory management in newer versions of SQL Server and Windows Server make it less likely that this setting will make a big difference.

2.14.2 Large Pages

For system with over 16GB of memory consider using Large Pages. The concept is to use a larger page size for memory as organized by the kernel. This makes the process of virtual address translation typically faster and should improve system performance when dealing with large amounts of RAM.

SQL Server uses Large Pages if:

- SQL Server Enterprise Edition
- The computer must have 8Gb or more of physical RAM
- The "Lock Pages in Memory" privilege is set for the service account.
- Trace Flag 834 is set
- You should only use Large Page on a system dedicated to SQL Server.
- ''max server memory' must be set to allow enough memory for the OS.

At start up the buffer is filled, so the system will take longer to start up. If you consider using this feature test it first to make sure you understand the impact on startup etc.

2.15 *Maximum Degree of Parallelism (MAX DoP)*

Siemens recommends that the Max Degree of Parallelism option in SQL Server be set to 4 and the cost threshold be set at 50. This will allow parallel queries to be used for some scanning queries, up to 4.

While the high-cost threshold will make it rarely used with the exception of certain use cases. This is preferable given the type of query workload that the Teamcenter application places on its database. This value is set in the Server Properties dialog box in SQL Server Management Studio.

2.16 *Delayed Transaction Durability*

SQL Server 2016 introduce a new option “Delayed transaction durability”. When enable the Transaction log records are kept in a buffer and written to disk when the buffer fills or a buffer flushing event takes place. Delayed transaction durability reduces both latency and contention within the system because:

- The transaction commit processing does not wait for log IO to finish and return control to the client.
- Concurrent transactions are less likely to contend for log IO; instead, the log buffer can be flushed to disk in larger chunks, reducing contention, and increasing throughput.

When to use it:

- **You can tolerate some data loss.** In general, the loss if any is minor and will not affect Teamcenter. You lose data held in the buffer; it is equivalent to the system crashing a few seconds earlier from the data loss perspective. If Teamcenter is logging some real time data. So, for 90% plus of Teamcenter system this is not an issue
- **You are experiencing a bottleneck on transaction log writes.**
  If your commits time slow, that is greater the a few milliseconds.
- **Your workloads have a high contention rate.**
  If your system has workloads with a high contention level much time is lost waiting for locks to be released. Delayed transaction durability reduces commit time and thus releases locks faster which results in higher throughput.

If you are using Always On or failover clustering some delayed durable transactions might be lost. Also, with Always On there is no guarantee of any durability on either the primary or any of the secondaries. In addition, they do not guarantee any knowledge about the transaction at the secondary. Teamcenter has additional support for Always On (12.0 onwards) which attempts to handle lost transactions when Always On fails over mid transaction. This has only been tested with standard durability.

3 Installing Teamcenter Databases

3.1 *File Placement on the Disk Subsystem*

Before installing the Teamcenter database, Siemens recommends planning the placement of the files that make up the database. In the default configuration, the Teamcenter database consists of two files: a data file and a log file. It is general SQL Server best practice to place these files on separate physical disk volumes to get the best performance from the overall system. These disk volumes should also not be used for any other heavily accessed files or for the Windows operating system.

It is critical that the disk volumes on which you place your database files be fault-tolerant RAID sets. The ultimate safety of your Teamcenter data depends on the data and log files residing on redundant physical disk storage. Siemens recommends RAID 10 (also called RAID 1 + 0) for the data file. RAID 5 is acceptable. For the log file, RAID 10 or RAID 1 is recommended. The log file should never be placed on a RAID 5 volume because RAID 5 has lower write performance than the other RAID levels. The following table shows a summary of recommendations for where to place the Teamcenter database files.

File Type	Placement	Recommended RAID Level	Disk Speed
Data (.mdf, .ndf)	Physically separate disks or LUN from the log Disk(s) should not be used for other heavy file I/O applications	RAID 10	10,000 RPM or faster SSDs are ideal
Log (.ldf)	Physically separate disks or LUN from data files Disk(s) should not be used for other heavy file I/O applications	RAID 1 or RAID 10 RAID 5 is discouraged for logs	10,000 RPM or faster SSDs are ideal

Note that when placing data and log files on separate disk volumes, they should be truly physically separate. The drive letters on which the files are placed should actually map to distinct physical disk drive sets. Placement on different partitions/LUNs on a single disk drive set is not helpful for performance or fault tolerance.

3.2 *Sizing Data Files Up Front*

Most Teamcenter installations will see significant database growth over time. For best long-term performance, Siemens recommends creating data and log files with a size that matches the long-term size expectations of the Teamcenter database.

If no explicit size is provided, SQL Server creates data and log files based on the size of a special template database called model. Usually, this is just a few MB in size. As data flows in to the Teamcenter database, SQL Server automatically extends the data and log files as needed. For large, heavily used databases, this eventually produces data files that are highly fragmented on disk. This fragmentation has a negative effect on performance. There is also a small run-time performance penalty every time a file is extended.

These problems can easily be avoided by sizing files appropriately when the database is created. The following guidelines are recommended for setting the initial size of database files.

1. Estimate long-term size of the Teamcenter database (e.g., 1-3 years).

2. Add 50% for small-sized databases (< 100GB). Add 35% for medium-sized databases (> 100GB). Add 25% for larger databases (> 500GB). This is the recommended data file size.

3. Size the transaction log file at approximately 20% of the data file size.

Automatic file growth should be enabled for the database, even though the initial size ought to be sufficient for some time. This is simply to avoid a production problem if the data files run out of space. As a rule, though, file size for the Teamcenter database should be managed proactively. Siemens recommends configuring data files for 5GB growth increments and log files for 1GB growth increments. In particular, avoid using large percent-based growth settings on large log files, because the initialization of new log segments can temporarily block activity on the database.

3.3 *Spread the PRIMARY Filegroup across Files on Multiple Disks (optional)*

For heavily used Teamcenter databases, overall performance can be affected by the throughput limitations of working with a single data file on a single disk volume. With SQL Server, you can easily spread I/O load for your Teamcenter database across multiple disk volumes or RAID sets by mapping SQL Server filegroups to multiple files. The extra administrative work of setting this up should only be done if you have confirmed that disk I/O is a bottleneck for your system. The monitoring techniques discussed later in this paper can help in making that determination.

If you choose to add additional files to spread I/O across disks, there are a few things to consider. First, Teamcenter creates all data objects (except optional audit logs) on the PRIMARY filegroup, so this is the one you will want to map to the new files. Second, SQL Server balances

I/O most effectively when the files used by a filegroup are of equal size. Try to create all the files that will host the PRIMARY filegroup so that they are the same size.

The Teamcenter database also contains a filegroup called ILOG. In the default database configuration, ILOG is hosted in its own file. Objects in the ILOG filegroup are created and used only if Teamcenter audit logging is enabled and configured to log to the database (rather than text files for older versions of Teamcenter). If your deployment is set up to log to the database, you can expect heavy activity on this file. **Therefore, if you use the database for audit logging, Siemens recommends placing the ILOG data file on a separate disk volume from the main data file and the transaction log.**

3.4 *Use Data Compression (Enterprise Edition only)*

SQL Server Enterprise Edition includes the ability to natively compress its data pages on disk. This can provide significant space savings on disk. Because it reduces the amount of disk I/O done for a given amount of Teamcenter data, it also tends to improve performance. When data compression is enabled, data pages are also compressed in the buffer pool (i.e., RAM). This provides the additional benefit of increasing the effective amount of application data that can be accessed from the data cache rather than disk.

Benchmarking with Teamcenter and SQL Server has shown significant performance increases when data compression is used for the Teamcenter database. Siemens recommends using data compression if your edition of SQL Server allows it.

4 Adjusting SQL Server and Database Settings

4.1 *Verify Database Collation*

When the Teamcenter database is created, it will assume the server collation (set when SQL Server was installed) unless otherwise specified. It is a good idea to verify that the collation for the database is the one the Teamcenter software expects. This can be verified in the Database Properties dialog box in SQL Server Management Studio. The generally requirement the collation should be "<locale>_BIN" where locale is SQL term for encoding you want for example Latin1_General_BIN. SQL Server installs it's base db's (master/tempdb/model) with CI/AS which is not an issue, it is only the Teamcenter database that has the case sensitive BIN requirement.

4.2 *Automatic Statistic Updates, Auto Shrink, and Other Options*

SQL Server uses internal statistics about the data stored in tables to decide which indexes, if any, to use for most efficiently executing a query. As table data changes over time, these statistics need to be kept current. Each SQL Server database has the option for these statistics to be updated automatically (statistics update can also be initiated manually, if needed).

Siemens recommends that auto-update of statistics be enabled for the Teamcenter database. Both Auto-Close and Auto-Shrink should be set to False.

4.3 *Choosing a Database Recovery Model*

The recovery model of a SQL Server database determines how changes are recorded and stored in the database's transaction log. The three options for the recovery model setting on your Teamcenter database are Full, Bulk-logged, and Simple:

- **Full recovery model**: This is the best model for preventing critical data loss and restoring data to a specific point in time, and it is generally used by enterprise production systems. If the transaction log is available, it is possible to get up-to-the-minute recovery and point-in-time restore if the end of the transaction log is backed up and restored. The trade-off for the Full recovery model is that it requires more administrator attention and a more complex backup/restore strategy.

- **Bulk-logged model**: This model is for databases that have critical data, but for which the DBA wants to minimize storage or performance impact of bulk operations (e.g., bulk data loads, table copies, index maintenance). It provides nearly the recovery power of the Full model, but with some limitations.

- **Simple recovery model**: This model is appropriate if the data backed up is not critical, data is static or does not change often, or if data loss is not a concern for the organization. If an active database is lost with the Simple recovery model, the organization loses all transactions since the last full or last differential backup. This model is typical for test environments or production databases that are not mission critical or have very light activity.

In general, Teamcenter databases are frequently updated and hold information that is of high importance to the organization. Especially for large Teamcenter databases, full backups may not be feasible with high frequency (e.g., daily or more often). For these reasons, Siemens recommends using the Full recovery model with Teamcenter databases. Administrators must be aware of the management implications of the Full and Bulk-logged recovery models. In particular, under these recovery models, transaction logs must be backed up regularly to prevent them from filling up or exhausting available disk space.

5 SQL Server in Azure Virtual Machines

5.1 *Overview*

This section contains information for optimizing SQL Server performance in the Microsoft Azure Cloud. While running SQL server in Azure it is recommended that you follow all the standard performance and tuning options that you would implement on a bare iron deployment. However, the performance of the database in a cloud environment depends on several key factors, such as the sizing of the VM, the configuration of the storage and the database workload.

5.2 *VM Size Guidance*

For performance sensitive applications such as Teamcenter it is highly recommended that you use a DS3_v2 at the very minimum or higher (such as an E series) at the time of this update. It is important to note that the "S" denotes support of the premium storage tier, which is recommended for best performance. There is always a tradeoff between cost and performance, so the specific VM size may vary depending on your individual workload requirements. Several VM configurations are available with constrained core sizes to reduce overall licensing cost. No matter the specific VM class you choose ensure that it supports the premium storage option.

5.3 *Storage Guidelines*

There are several configurations that support the premium SSD option, this is recommended for all production instances. The standard SSD options have varying latencies and bandwidth and are only suitable for test/dev workloads.

5.4 *Disk Guidance*

There are three main disk types offered in Azure:

- OS Disk: When you create an Azure instance, there will be at least one disk mounted as the C drive, this is the operating system drive for the VM. This disk is a VHD stored as a page blob in storage.
- Temporary Disk: An Azure VM also has a drive mounted as the letter D, this is a disk on the node that can be used for scratch space.
- Data Disk: You can also attach additional disks to your VM as data disks, and these will also be stored in storage as a page blob.

The OS disk should be used for system files only, the default caching policy on the OS disk is Read/Write. For performance sensitive applications such as Teamcenter you should install the database on a data disk.

The Temporary disk is not persistent! Do not store any of your user database files or logs in the temp disk! It is however possible to store TempDB files on the D drive and may result in higher

throughput and lower TempDB latency. However, there is an exception to this, if your workload profile on TempDB is very write intensive you can achieve best performance by locating it on premium SSD storage instead.

Data disks should be used for your data and log files. If you are not using disk striping, use two premium SSDs with one containing the data and TempDB and the other used for log files. If additional throughput is required, you can add additional data disks and use disk striping. Before implementing this, you will need to analyze the number of IOPs, and bandwidth required for your log and data files. Be aware that different VM sizes come with different IOP limits.

5.5 *Caching Policy*

The following recommendations only apply when using the premium or Ultra SSDs, if you are using standard SSDs do not enable any caching.

- If you are using separate disks for data and logs, enable read caching on the disks hosting your data and TempDB files. This can greatly increase performance, do not however enable this for the disk holding the log files as it will cause a decrease in performance.
- If you are using striping in a single storage pool most workloads will benefit from read caching. If you have separate storage pools for the log and data files, only enable read caching for the data files. In certain very write heavy workloads you may achieve better performance without having caching enable. However, this can only be determined through testing.

5.6 *NTFS Allocation Size*

When formatting the data disk use the 64-KB allocation size, this applies to disks which will store the data, log and TempDB files.

5.7 *I/O Guidance*

- Database page compression can help improve the performance of I/O intensive workloads. However, this comes at the cost of higher CPU consumption on the server.
- Use instant file initialization to reduce the time it takes for file allocation. To do this you must grant the SQL Server service account with SE_MANAGE_VOLUME_NAME and add it to the Perform Volume Maintenance Tasks security policy. By default, on Azure the service account is not included in that security policy.
- Autogrow should not be used to manage your data and log growth, it was designed to handle unexpected events. If you use autogrow then you should pre-grow the files using Size switch.
- Ensure that Autoshrink is disabled to avoid overhead that may impact performance.
- Enable “Lock Pages” in memory privilege to reduce I/O and paging activities.

- If you are using SQL Server 2012, ensure that you are at a minimum of Service Pack 1 Cumulative Update 10.

6 Ongoing Management

6.1 *The Database Backup Plan*

Perhaps the single most important ongoing maintenance task for a business-critical database such as Teamcenter is managing backups. Siemens recommends you carefully plan, document, and test the backup and recovery strategy for the Teamcenter database.

The types of database backups allowed in SQL Server are:

- **Full database backup:** Contains all the data in the database, plus enough transaction log information to allow for recovering that data.

- **Differential database backup:** Contains all the data changed since the last full backup. Differential backups allow a smaller, faster backup to be taken, which can later be combined with the full backup to restore the database to the time of the differential backup.

- **Transaction log backup:** Contains all the database changes since the last (full or differential) database or log backup. Periodically backing up the log allows you to restore the last database backup followed by subsequent log backups to recover to the latest point in time needed.

- **Database file or filegroup backup:** Contains only data from specified file(s) or filegroups(s) to allow quick, piecemeal restores if only part of the database is lost. These types of backup are not recommended for Teamcenter deployments.

The different types of backup allow you to back up with the frequency appropriate to your business requirements, even for databases that are very large. For example, a typical backup strategy for a large database (e.g., several hundred gigabytes) might include a weekly full backup, nightly differential backups, and transaction log backups every 30 minutes. If the database were lost, the DBA would:

1. Restore the last full backup.
2. Restore all differential backups since the full backup, in sequence.
3. Restore all the transaction log backups since the last differential backup, in sequence.

If properly executed, this would return the database to operation with at most 30 minutes of work lost.

For small databases, it is perfectly valid to simply perform full backups one or more times per day. There is nothing inherently wrong with this approach, and it has the virtue of simplicity. But it is only practical for databases that can be backed up very quickly.

Keep in mind also that for databases with the Full or Bulk-logged recovery model, regular transaction log backups are required to keep the log from growing out of control.

The right backup strategy is dependent on the database size and level of change activity in your installation. This can vary greatly, even across deployments of the Teamcenter product, so a one-size-fits-all recommendation can't be made here. However, keep the following in mind as you plan your backup strategy:

- If you choose the Full recovery model, the transaction log can grow very quickly and should be scheduled for backup at multiple times throughout the day. Performing multiple backups ensures that the log is truncated frequently and provides for improved data recoverability.

- If you choose the Simple recovery model, schedule full database backups for at least once per day. With this method, data recoverability is limited to the last full database backup.

- Database backups should be stored off site when possible and appropriate.

- Back up a database immediately after you make extensive changes or perform non-logged operations.

- Be sure to include regular backups of the master and msdb databases as part of your overall backup strategy. The master database contains information about the databases on your system, and the msdb database stores information about scheduled jobs.

- Use backup compression to reduce the size of backups and the time required to take them.

Finally, remember that what you really need in an emergency is a restore, not a backup. Therefore, plan restore with care, and test restoring your database from actual backups to validate that you have covered everything.

6.2 *Synchronization of Teamcenter Database and File Backups*

Another point to consider when managing overall backup of the Teamcenter application is the interaction between the SQL Server database and the file repository. The Resource Tier includes both the Teamcenter database and the Teamcenter file server(s). A full and consistent backup of the Resource Tier requires both database and file backups.

The Teamcenter database contains references (essentially file pointers) to content in the file repository. To ensure maximum consistency between the database file references and the

actual content of the file repository, some attention must be paid to backing up both simultaneously.

Siemens recommends choosing one of three approaches to doing this:

- **Option 1:** Take cold backups—that is, take the system offline to halt user activity, back up both the database and file repository, and put the system back online.

- **Option 2:** Take online backups and accept some small potential inconsistency. The issues with this approach can be minimized if the backups are taken during periods of low activity.
- **Option 3:** Use Teamcenter's hot backup mode. In this mode, Teamcenter suspends writing files directly to the file repository (they are held in a temporary location called a Blobby Volume). File backups are taken, and the system is then returned to normal mode. If a restore is necessary, SQL Server's point-in-time restore capability can be used to restore the database to the point in time when the system was placed in hot backup mode. This will ensure consistency between the database and the file repository.

More information on this advanced topic is available from Siemens. You can talk with your Siemens support professional or refer to documentation at the Siemens Teamcenter Support Center.

6.3 *Monitor Index Fragmentation and Defragment When Necessary*

SQL Server uses indexes to provide fast access to information when users or applications request it. These indexes are maintained by the Database Engine as the table data grows and/or changes. Over time, the indexes can become fragmented; especially in databases that handle heavy insert, update, and delete activity. An index is "fragmented" when the physical ordering on disk does not match the logical order of the data (as defined by the index key) or when data pages that contain the index are dispersed across non-adjacent sections of the disk.

Fragmentation of an index can reduce the speed of data access and result in slower application performance. It can also cause more disk space to be used than is necessary. Index fragmentation can be corrected by reorganizing or rebuilding the index.

You can tell which indexes, if any, have fragmentation problems by using the sys.dm_db_physical_stats() system function. This function provides lots of details about the physical layout of the index. However, the most important result column for tracking fragmentation is avg_fragmentation_in_percent. This column indicates how fragmented the index is on disk. A low number means low fragmentation (good); a high number means high fragmentation (bad). Similar information is also available in the Standard Reports in SQL Server Management Studio.

The following table indicates general guidelines for interpreting the avg_fragmentation_in_percent value. A fragmentation report is also available by utilizing the Teamcenter utility index_verifier, note the fragmentation report is only available when using MS SQL Server.

Fragmentation	Recommended Action
< 5%	Do nothing
5% to 30%	Reorganize with ALTER INDEX REORGANIZE
> 50%	Rebuild with ALTER INDEX REBUILD WITH (ONLINE=ON) or CREATE INDEX with DROP_EXISTING=ON

Reorganizing an index does not block user access to the index while underway. However, rebuilding or re-creating the index does prevent user access to the index. The exception to this is if the ALTER INDEX REBUILD statement is used with the ONLINE = ON option. Note that online index rebuild requires the Enterprise Edition of SQL Server.

Periodically checking index fragmentation and taking any necessary corrective action is important to maintaining the performance of your Teamcenter deployment. The rate at which fragmentation may occur depends on user activity, but as a general rule, Siemens recommends checking index fragmentation at least monthly.

6.3.1 Script Resources

There are several online resources for scripts to assist in these maintenance activities. One popular resource is Ola Hallengren's website https://ola.hallengren.com/sql-server-index-and-statistics-maintenance.html which provides excellent examples.

Note: There is a problem with 11.2.1 fixed in 11.3. POM_M_LOCK has an index REF_UID and ASPACE_UID, this causes plan problems. So, drop index PIPOM_M_LOCK;
create index PIPOM_M_LOCK on POM_M_LOCK("REF_UID"); (PR 8005808)

6.4 *Run the Teamcenter Index_Verifier Tool*

Siemens provides an index_verifier utility, packaged with all versions of Teamcenter that can analyze your Teamcenter database for missing indexes. If missing indexes are found, the index_verifier utility will provide SQL output that you can use to create the needed indexes.

Siemens best practice is to run this utility at least monthly.

Siemens also recommends adding additional indexes you may require by using the install command-line utility provided with Teamcenter. Doing so will register the index as part of the Teamcenter schema, at which point the index_verifier utility can check for its existence. If you

manually add indexes to the Teamcenter database, index_verifier will have no knowledge of them, and it will be your responsibility to ensure that those indexes are carried forward and maintained.

Full documentation of these utilities can be found in the Teamcenter Utilities Reference Manual, available on the Siemens Teamcenter Support Center.

6.5 *Periodically Check Database Integrity*

For all the sophisticated data management techniques embedded in the SQL Server Database Engine, there is still the possibility of some corruption occurring in a database, most notably as the result of a hardware glitch. To head off the potential impact of such problems, you should regularly check the integrity of the database. The statement for doing this in T-SQL is DBCC CHECKDB.

It is best to include database integrity checks in a scheduled job that executes DBCC CHECKDB or do it with a scheduled Maintenance Plan that includes the Check Database Integrity task.

If any problems are detected, you can restore from backups (usually the best option) or use one of the REPAIR options on DBCC CHECKDB.

6.6 *Monitor Space Used*

Over time, the space used in your Teamcenter database for data and indexes will increase. Regardless of the initial space allocations you have made, you will need to occasionally check on the amount of space remaining in data and log files as well as on the host disk volumes.

Space consumed and remaining for the Teamcenter database can easily be checked using the Standard Reports in SQL Server Management Studio. The Disk Usage report displays the currently reserved space in data and log files and the amount that is in use. This is also a handy place to check for any recent data or log file growths. If these are occurring, it could be a sign that a rearrangement or expansion of data files is necessary.

Details on space usage can also be obtained via T-SQL by using the sp_spaceused stored procedure.

6.7 *Use SQL Server Agent Jobs and Alerts*

Many of the ongoing database management and monitoring tasks required to administer your Teamcenter database can and should be automated. SQL Server includes SQL Server Agent, a job scheduling and alerting service that is tightly intertwined with the SQL Server database engine. SQL Server Agent can be used to schedule regular backups, database integrity checks,

index reorganization jobs, and many other routine tasks. It can also be used to check database engine metrics and alert someone if they are outside normal bounds.

SQL Server Maintenance Plans also provide a neatly packaged way to define and schedule routine management tasks for a database or for multiple databases.

Siemens recommends making use of these tools to configure automated maintenance and monitoring tasks for the Teamcenter database.

7 Tools for Monitoring Database Server Performance

Once your Teamcenter database is in production, it is important to establish a baseline of its normal performance characteristics and to check its health and status periodically. The following sections discuss some of the tools and techniques available for monitoring database server performance.

7.1 *SQL Trace*

SQL Trace is a server-side, event-driven activity monitoring tool; It can capture information about more than 150 event classes. Each event returns data in one or more columns and you can filter column values. You configure the range of events and event data columns in the trace definition.

You can also configure the destination for the trace data, a file or a database table, in the trace definition. You can use it to:

- Understand the duration, frequency, and resource usage of queries.
- Gather a baseline or create a benchmark for a system's activity.
- Troubleshoot application errors and performance problems.
- Audit user activity.

SQL Trace is included in SQL Server 7.0 and later versions.

7.2 *SQL Server Profiler*

SQL Server Profiler is a GUI for creating SQL traces and viewing data from them. SQL Server Profiler is included in SQL Server 7.0 and later versions.

Important Note: SQL Trace and SQL Server Profiler have been marked for deprecation since SQL Server 2012. SQL Server 2016 includes SQL Server Profiler and SQL Trace, but Microsoft has declared an intention to remove both tools in a future version of SQL Server. Extended Events is now the recommended activity tracing tool.

7.3 *Extended Events*

Like SQL Trace, Extended Events is an event-driven activity monitoring tool; however, it attempts to address some of the limitations in the design of SQL Trace by following a loose-coupled design pattern. Events and their targets are not tightly coupled; any event can be bound to any target. This means that data processing and filtering can be carried out independently of data capture. In most cases, this results in Extended Events having a lower performance overhead than an equivalent SQL Trace.

With Extended Events, you can define sophisticated filters on captured data. In addition to using value filters, you can filter events by sampling and data can be aggregated at the point it is captured. You can manage Extended Events either through a GUI in SQL Server Management Studio (SSMS) or by using TransactSQL statements.

Extended Events was introduced in SQL Server 2008; since the deprecation of SQL Trace and SQL Server Profiler was announced with the release of SQL Server 2012, many features introduced in SQL Server 2012, 2014 and 2016 can only be traced using Extended Events.

Extended Events replaces SQL Server Profiler and SQL Trace. You can therefore use it to perform regular monitoring, such as the following:

- Troubleshoot blocking and deadlocks.
- Find resource-intensive queries.
- Capture dynamic-link library (DDL) statements.
- Capture missing column statistics.

You can also use Extended Events to collect information that you were not able to monitor with SQL Trace. You can now:

- Observe page splits.
- Monitor session-level wait statistics.
- Monitor stored procedures that exceed the last CPU, I/O, and execution times.
- Observe the proportional fill algorithm that is used to fill data in data files.

Although Extended Events is a lightweight logging framework, each active session has an overhead of CPU and memory resources. You should get into the practice of only running Extended Events sessions you have created when you have to troubleshoot specific issues.

7.4 *SQL Server Activity Monitor*

A good, easy-to-use starting point for monitoring your SQL Server instance is the Activity Monitor in SQL Server Management Studio. This pulls several key pieces of information together into one view. Although it does not provide highly detailed insight on its own, it can be a good part of your SQL Server monitoring toolbox.

To open Activity Monitor, right-click the server icon in the Object Explorer pane of SQL Server Management Studio and select the Activity Monitor item. There is also a button for Activity Monitor on the Standard toolbar in SQL Server Management Studio. The Activity Monitor will open as a new tab in the main Management Studio window.

The Overview section, at the top, shows summary graphs of four key metrics for getting a sense of the load on the server. % Processor Time is an average of CPU utilization across all processors. Waiting Tasks shows the number of tasks waiting on CPU, memory, disk, locks, or other resources. The Database I/O graph shows overall disk I/O throughput of SQL Server. Batch Requests/sec shows the number of individual T-SQL batches executed per second (e.g., SQL statements or stored procedures calls from applications).

The other sections (also called panes) in the Activity Monitor window provide detailed snapshots about different aspects of the Database Engine's current workload. A full discussion of these sections and how to interpret the information provided is well beyond the scope of this paper. However, a brief introduction to each section is provided here.

The Processes section presents a list of the active user connections to the server. For each connection, useful troubleshooting information is provided, including the current database the connection is using, what resource the connection is waiting on (if any), and whether the connection is blocked by another connection. The Processes section is thus useful for getting an idea of the number of active connections and what sort of performance roadblocks they may be experiencing.

The Resource Waits section provides a summary of the "waits" the Database Engine is observing in current activity. Examples of waits are network I/O for clients to pull results across the wire, disk I/O for the server to pull data from disk into buffer memory, or lock waits while one connection waits on a transaction lock held by another connection. Besides the type of wait, this section shows the cumulative and recent amount of time spent on this type of wait and the number of connections experiencing it. This information is just a coarse summary of a highly detailed set of tuning information available from SQL Server wait statistics. It can be quite useful, though, to get an indication of system bottlenecks at a glance.

The Data File I/O section lists all data and transaction log files in use by databases on the current SQL Server instance. With each file, recent read and write throughput (in MB/sec) is reported. The last column in this section is particularly interesting. The Response Time (ms) column reports disk latency for I/O against the file. This is a good number to look at to begin an investigation of whether a workload is I/O bound.

Finally, the Recent Expensive Queries section lists some of the most expensive queries the engine has recently executed. Besides the query text, average execution metrics are listed for each.

7.5 *Standard Database Reports in Management Studio*

Another easy-to-use resource is the set of database Standard Reports in SQL Server Management Studio. These reports are accessible by right clicking the icon for the Teamcenter database in the Object Explorer pane and selecting Reports > Standard Reports. The reports available on the fly-out menu provide basic and advanced information on many aspects of the database.

7.6 *Performance Monitor*

Performance Monitor is a Windows management snap-in you use to analyze system performance. It can be used for real-time monitoring of current system performance and to log data to be analyzed later. It provides different counters that are categorized into objects to monitor Windows and other applications that are running on the Windows operating system.

SQL Server makes a large number of performance counters available to Performance Monitor to make it easier for you to monitor SQL Server activity and Windows activity from one place. Some examples of SQL Server performance counters that are typically used when examining SQL Server performance are:

- SQL Server: Buffer Manager: Buffer Cache Hit Ratio
- SQL Server: Buffer Manager: Page Life Expectancy
- SQL Server: SQL Statistics: Batch Requests/Sec
- SQL Server: SQL Statistics: SQL Compilations/Sec
- SQL Server: SQL Statistics: SQL Recompilations/sec

Note: In most cases it is expected that you will see page life to be very long, in some cases days if the system has a large amount of memory. However, you may see drops when various administrative functions are performed. If you see drops in life expectancy during periods of high database utilization, then that indicates that you need more memory in the system.

It is good practice to establish a baseline understanding of the performance of your database system after it is deployed. This will allow you to better understand issues if your system runs into performance or other problems later. By comparing observed behavior in a problem situation to your baseline expectations, you'll more easily be able to focus in on the root cause.

Note that if you are using a named instance of SQL Server, the name of the object in Performance Monitor will be slightly different. For default (not named) instances, the object name is SQL Server: <object>. For a named instance, the object name is MSSQL$InstanceName: <object>.

7.7 *PAL (Performance Analysis of Logs)*

PAL is not a tool that ships with SQL Server, rather it is an independently developed utility available for free that will analyze any metrics gathered with the Performance Monitor. PAL can be configured to analyze SQL specific metric data via templates and output a report in HTML format. For more information on PAL including current versions search for the project on GitHub.

7.8 *Dynamic Management Objects*

Dynamic Management Views (DMVs) and Dynamic Management Functions (DMFs) can provide insight into the internal state, health, and performance of SQL Server through Transact-SQL queries. They provide useful information that can be used to identify, diagnose, and fix SQL Server performance issues from counters and metrics that SQLOS collects. Each Dynamic Management Object (DMO) will return data either at server scope or database scope.

There are more than 200 DMOs in SQL Server 2016. DMOs can be found in the sys schema; their names start with “dm”. They are further divided into other categories based on the kind of information that they return. Some of the most useful DMO categories that you can use when performance tuning SQL Server are listed here:

- sys.dm_exec_*. This set of DMVs contains information that relates to the execution of user queries. They are useful in diagnosing blocking, deadlock, long running queries, CPU intensive queries, and so on. For example, sys.dm_exec_requests returns information about currently executing requests.

- sys.dm_os_*. This set of DMVs gives details about SQLOS operations; that is, memory management and scheduling. For example, sys.dm_os_wait_stats returns aggregate information about all of the waits that have been incurred on an instance of SQL Server. The sys.dm_os_schedulers DMV returns one row per scheduler and is used to monitor the health and condition of the scheduler, or to identify runaway tasks.

- sys.dm_tran_*. This set of DMVs provides details about current transactions and isolation. For example, the sys.dm_tran_active_transactions DMV returns transaction details for a SQL Server instance.

- sys.dm_io_*. This set of DMVs provides insight into the I/O activity in SQL Server. For example, sys.dm_io_pending_io_requests identifies pending I/O requests that SQL Server has initiated.

- sys.dm_db_*. This set of DMVs provides details about the database and database-level objects such as tables and indexes. For example, sys.dm_db_index_physical_stats is a DMF that returns fragmentation information across all indexes in a database or for a specific index.

For a complete list of DMOs in SQL Server 2016, see the topic Dynamic Management Views and Functions (TransactSQL) in SQL Server Technical Documentation.

7.9 *Database Engine Tuning Advisor*

The Database Engine Tuning Advisor is a graphical and command-line tool that analyzes how the SQL Server Database Engine processes queries and makes recommendations for improving the performance of queries through the creation of indexes, partitions, indexed views, and statistics. It is provided with SQL Server and can consume a trace file, analyze the queries that are captured, and then provide details of:

- New indexes and statistics that can increase query performance.
- Suggest partition strategy.
- The usage of existing indexes.

7.10 *Diskspd*

Diskspd is a command line utility, developed by Microsoft, for load generation and performance testing of storage I/O subsystems. Although Diskspd is not specifically designed for simulating SQL Server I/O activity, it can be configured to do so. Diskspd replaces SQLIO, an older utility that had a similar purpose.

Diskspd is suitable for use when performance tuning I/O subsystems because, for a given set of parameters, the load that is generated will always be the same.

Diskspd is available as a download from Microsoft. Detailed instructions on simulating SQL Server I/O activity, and instructions on how to interpret the results, are included in a document that is packaged with the Diskspd download (UsingDiskspdforSQLServer.docx)

7.11 *Live Query Statistics*

When working with SQL Server 2016, a new feature called Live Query Statistics offers a dynamic way to view the actual execution plan for a query. When Live Query Statistics is enabled (on the Query menu, click Include Live Query Statistics), counts of rows and other properties of the actual execution plan are displayed and updated in real time.

Note: Be aware that gathering the information required to run Live Query Statistics will place additional load on your SQL Server instance.

8 Known Issues

8.1 *Ghost records*

- Records that have been logically deleted but physically exist on a page.
- Simplifies keyrange locking and transaction rollback.
- A record is marked as deleted by setting a bit. A ghost cleanup process physically deletes the ghost records after the calling transaction commits.

8.1.1 How to prevent Ghost Records from being an issue

See Section 20.10.2 for details

8.2 *Cardinality Estimator Bug*

SQL Server 2014 introduced a new Cardinality Estimator (CE) which can cause Teamcenter problems.

- If the SQL Server installation is upgraded the database compatibility is kept the previous level and you should not see an issue.
- If the SQL Server is a new install and a database import the compatibility is at the new level
- If the system experiences performance issues you can:
- Revert to the old CE

```
ALTER DATABASE
   SCOPED CONFIGURATION
      SET LEGACY_CARDINALITY_ESTIMATION = ON;
 go
```

- Or revert to the old optimizer

```
ALTER DATABASE <yourDatabase>
   SET COMPATIBILITY_LEVEL = 110;
go
```

This still seems to still be an issue in SQL Server 2016 and should be fixed in 2019 CU 4 and 2017 CU 19.

Note: Also set TC_ODBC_NO_PREPARE=True to prevent pointless additional database calls. This is defaulted on at 11.2.3

8.3 *Lock Escalation*

When using the new locking enhanced mechanism delivered 11.6.0.7, 12.1.0.6 and 12.2 onwards lock escalation on the POM LOCK tables has caused blocking locks. Typically, this has been caused by batch processes and dispatcher.

Analysis by development has shown the TIMESTAMP table could also be a subject to lock escalation causing a blocking lock.

User of Qsearch users may also have this issue with PQS_PRODUCT_PATH_INDEX

Disable lock escalation on the LOCK tables etc., using the following SQL or via The Management Studio GUI. The overhead on disabling lock escalation on these tables is a small increase in memory usage.

```
ALTER TABLE POM_M_LOCK SET (LOCK_ESCALATION = DISABLE);
ALTER TABLE POM_R_LOCK SET (LOCK_ESCALATION = DISABLE);
ALTER TABLE POM_F_LOCK SET (LOCK_ESCALATION = DISABLE);
ALTER TABLE POM_TIMESTAMP SET (LOCK_ESCALATION = DISABLE);
ALTER TABLE PQS_PRODUCT_PATH_INDEX  SET (LOCK_ESCALATION = DISABLE);
ALTER TABLE SCRATCH_TABLE SET (LOCK_ESCALATION = DISABLE);

go
```

9 Checklist

For convenience, the following checklist summarizes and consolidates the best practices discussed in this paper. After familiarizing yourself with the preceding in-depth discussion, you can use this as a reminder to cover all the recommended best practices. Use a dedicated database server and instance.

	Use a dedicated database server and instance.
	Choose the right SQL Server Edition (Standard or Enterprise) for your needs.
	Use 64-bit SQL Server if your hardware allows.
	Choose a high availability strategy.
	Use a low-privilege Windows account for the SQL Server and SQL Agent services.
	Grant the Perform volume maintenance tasks permission to the SQL Server service account to allow fast file initialization.
	Set the collation to Latin1_General_BIN when installing SQL Server.
	Set the server Authentication Mode to Mixed.
	Configure the SQL Server Database Engine service and the SQL Server Agent service to Auto-Start.
	Size and place tempdb per best practices (as described in detail in this document).
	Set the maximum server memory for SQL Server per recommendations.
	Set Maximum Degree of Parallelism to 4 and the cost threshold to 50.
	Enable the Query Store
	Place the data and log files on physically distinct disk volumes.
	Use RAID 10 for data files; use RAID 10 or RAID 1 for log files.
	Use large initial size for data and log files, based on projected growth.
	Leave automatic file growth enabled
	Spread the PRIMARY filegroup across equal-sized files on multiple disk volumes (optional).
	Use SQL Server data compression (SQL Server Enterprise Edition only).
	Verify that the Teamcenter database is using the Latin1_General_BIN collation.
	On the Teamcenter database, use the following settings: • Auto Create Statistics = True • Auto Update Statistics = True • Auto Close = False • Auto Shrink = False Use the Full recovery model for the Teamcenter database.
	Enable SNAPSHOT ISOLATION and enable Teamcenter AUTO COMMIT
	Use the Full recovery model for the Teamcenter database.
	Ensure frequent transaction log backups are scheduled to prevent the log file from growing out of control.
	Plan and implement an effective backup strategy.
	Test database restore using live backups.

	Be sure to back up the master and msdb databases in addition to the Teamcenter database.
	Monitor index fragmentation, and defragment when necessary.
	Check database integrity before backups to ensure the data will be good to restore.
	Monitor space used.
	Use SQL Server Agent jobs and alerts to automate and schedule maintenance and monitoring.
	Run the index_verifier monthly.
	Get familiar with SQL Server monitoring tools and establish a baseline performance profile for your system.

10 Links for Further Information

- SQL Server Index and Statistics Maintenance (hallengren.com)
- Planning a SQL Server Installation - SQL Server | Microsoft Docs
- What's new in SQL Server 2019 - SQL Server | Microsoft Docs
- Troubleshoot slow SQL Server performance caused by I/O issues - SQL Server | Microsoft Docs
- Recommendations to reduce allocation contention - SQL Server | Microsoft Docs