SolidPractices:

SQL Server Performance

SOLIDWORKS® PDM

Last Update: April 2020

Revision 1.0

Table of Contents

Revision History

Rev #	Date	Description
1.0	*Apr 2020*	*Document updated for the latest software version. Revised for use by customers and reset as document version 1.0*

Note

All SolidPractices are written as guidelines. It is a strong recommendation to use these documents only after properly evaluating your requirements. Distribution of this document is limited to Dassault Systèmes SolidWorks employees, VARs, and customers that are on active subscription. You may not post this document on blogs or any internal or external forums without prior written authorization from Dassault Systèmes SolidWorks Corporation.

This document was updated using version SOLIDWORKS 2020 SP02. If you have questions or need assistance in understanding the content, please get in touch with your designated reseller.

1) Preface

This SolidPractice document describes various factors that influence the performance of Microsoft® SQL Server® as a component of a SOLIDWORKS PDM environment. The document discusses aspects that you can influence when configuring and maintaining a SQL Server installation (a SQL Server), and the best ways to identify and address performance problems.

It is difficult to assign rigid rules in this area. However, a good understanding of all available options is helpful when defining the best strategy for your specific requirements.

The expectation is that this document will be expanded and enhanced over a number of versions, as field experience provides feedback on the most useful areas.

Unless otherwise noted, specific commands and screen images in this document refer to SQL Server 2014. In most cases, the operations also apply to SQL Server 2017 or newer, although some details may vary.

The topics cover mainly the SQL Server Standard edition when used with SOLIDWORKS PDM Professional. When necessary, there are additional details about the SOLIDWORKS PDM Standard and the SQL Server Express and Enterprise editions.

Your Feedback Requested

We would like to hear your feedback and your suggestions for new topics. After reviewing this document, please take a few minutes to fill out a *brief survey*. Your feedback will help us plan and deliver content that directly addresses your challenges.

2) SOLIDWORKS PDM Architecture

Troubleshoot SQL Server performance if you are familiar with the SOLIDWORKS Product Data Management (PDM) architecture. For example, most issues reported involve SOLIDWORKS PDM applications. If you understand how those applications work, it becomes easier to determine if a problem actually relates to SQL Server and if following the current document guidelines are appropriate.

SOLIDWORKS PDM uses a server/client application based on the SQL Server that hosts the application's relational database. This means that SQL Server performance can influence the performance of all SOLIDWORKS PDM server services and client applications.

The SOLIDWORKS PDM main server components include the Database Server Service, the Archive Server Service and SolidNetWork Licensing (SNL). Optionally, components can include a Web Server Service or, the SOLIDWORKS PDM Web2 client (effective with SOLIDWORKS 2015 SP3) that you configure using Microsoft Internet Information Services (IIS). The Web2 client enables application access from multiple web browsers and mobile devices.

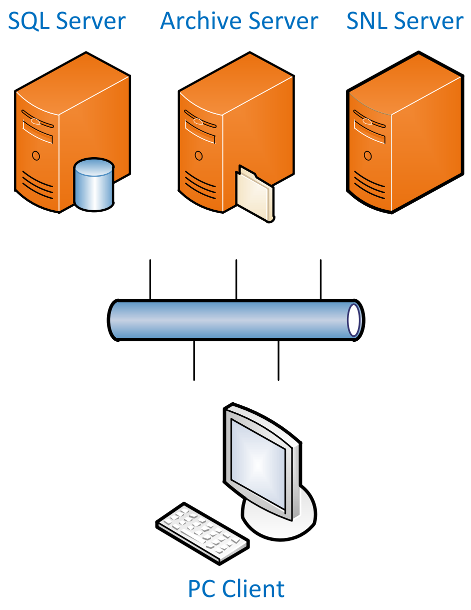


Figure 1: Minimal Setup

A SOLIDWORKS PDM setup can vary between simple configurations, as shown in *Figure 1*, to more complex configurations like those shown in *Figure 2*, where SOLIDWORKS PDM is in use with database replication and solutions such as Web2 and the EXALEAD® OnePart business discover application. There are also cases where a production environment has additional add-ins for Enterprise Resource Planning (ERP) integration or for other third-party applications.

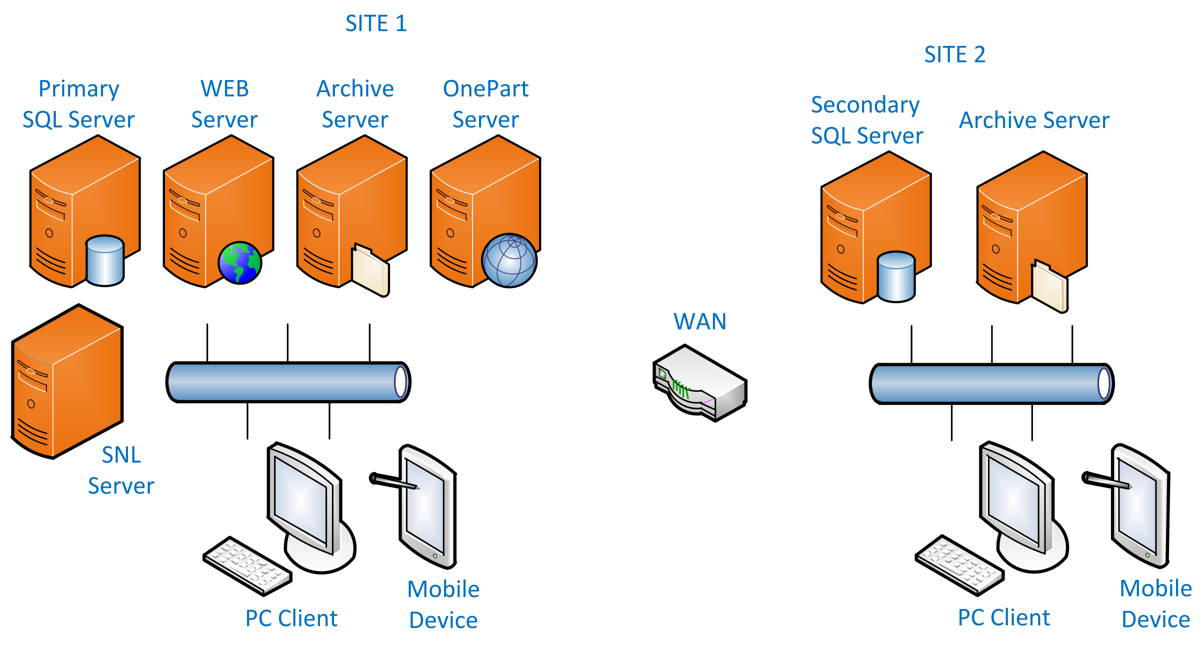


Figure 2: Complex Setup

The main SOLIDWORKS PDM client application is SOLIDWORKS PDM File Explorer, which is built on top of Windows® Explorer. Computer-aided design (CAD) applications such as SOLIDWORKS, DraftSight®, AutoCAD®, or Inventor® have special add-ins that enable the applications to access SOLIDWORKS PDM from within a CAD session. There are other client applications for vault administration and configuration of the archive server.

This SolidPractice makes frequent reference to SOLIDWORKS PDM components without providing much detail. Therefore, having a good knowledge of SOLIDWORKS PDM architecture is helpful to having a better understanding of the guidelines in this document. To learn more about SOLIDWORKS PDM components, read the SolidPractice: "*SOLIDWORKS PDM Professional Architecture,*" which is available on the VAR Resource Center (VRC) website under **Support** > **SolidPractices**.

a) Additional Reading Material

There are many articles about SQL Server performance in the Microsoft Knowledge Base (*http://support.microsoft.com*) and elsewhere. The techniques described in this document to improve SOLIDWORKS PDM performance are based on generally accepted SQL Server best practices. However, the converse is not necessarily true, in that not all generic SQL Server performance best practices are beneficial to a SOLIDWORKS PDM system.

3) Installation and Configuration

a) Overview

SQL Server performs best when using a dedicated server computer. As a best practice, do not install other applications on the SQL Server computer. An exception to this recommendation is the SOLIDWORKS PDM Database Server Service, which typically runs more efficiently when installed on the same computer as SQL Server.

The following sections examine how the various aspects of SQL Server influence performance.

b) Server Sizing

SQL Server available resources can influence the performance of SOLIDWORKS PDM client applications. This is especially true over time as production SQL databases increase in size. As this occurs, adjustments to the server configuration or resources may become necessary. For implementations with tens or hundreds of active users, it is advisable to have a server setup that allows an increase to the available CPU and memory. Most of the *supported virtual environments* have options to adjust these resources dynamically.

It is not advisable to set up a new SQL Server based only on the minimum system requirements. Instead, the setup should take into consideration the actual SQL workload for CPU sizing, and the SQL database size increase trend for memory. After monitoring usage of the main resources over one month of usage (see *"How to make performance baselines"* later in this document), you can adjust the available resources as necessary.

The recommendation is to monitor resource usage at least once every year or when there is a major change to the production environment. For example, when adding many new PDM users, when using a new third-party PDM add-in, or after activating replication for the database server or the archive server.

c) Operating System

SQL Server performs best when installed on a 64-bit server class of operating system. Support for Windows Server® operating system (OS) versions varies depending on which SOLIDWORKS PDM and SQL Server versions are in use. The following table depicts OS support for recent versions of SOLIDWORKS PDM.

	SOLIDWORKS Version				
Operating System Version	2016	2017	2018	2019	2020
Windows Server 2019	✗	✗	✗	✓ 2019 SP3	✓
Windows Server 2016	✗	✓ 2017 SP2	✓	✓	✓
Windows Server 2012 R2	✓	✓	✓	✓	✗
Windows Server 2012	✓	✓	✓	✗	✗
Windows Server 2008 R2, SP1	✓	✓	✗	✗	✗

With this in mind, you can determine which SQL Server platform supports your SOLIDWORKS production environment. The following table defines SQL support for recent SOLIDWORKS versions.

	SOLIDWORKS Version				
SQL Server Platform	2016	2017	2018	2019	2020
SQL 2017	✗	✗	✓	✓	✓
SQL 2016	✗	✓	✓	✓	✓
SQL 2014	✓	✓	✓	✓	✓ SQL 2014 SP3
SQL 2012	✓	✓	✓	✓	✗
SQL 2008 R2	✓	✓	✗	✗	✗

You should also consider operating system support for the SQL Server versions. For example, SOLIDWORKS PDM 2017 supports both Windows Server 2016 and SQL Server 2012. However, as shown in the following table, these two do not appear as a supported configuration.

SQL Platform	Windows Server 2008/2008 R2	Windows Server 2012/2012 R2	Windows Server 2016
SQL Server 2017	✗	✓	✓
SQL Server 2016	✗	✓	✓
SQL Server 2014	✓	✓	✓
SQL Server 2012	✓	✓	✗
SQL Server 2008 R2	✓	✓	✗

For an up-to-date list of system requirements, go to the SOLIDWORKS website, and navigate to **Support** > **Hardware & System Requirements** > **SOLIDWORKS Product Data Management**.

d) Recovery Model

When you create a database, SQL Server defaults to the **Full** recovery model. When using the SQL Server Standard edition, the recommendation is to switch the default to **Simple** recovery for a SOLIDWORKS PDM database. Be aware that when using database replication, the recovery model must remain as **Full** mode.

For more information about recovery, see the following solution in the SOLIDWORKS Knowledge Base:

S-029263   What is the difference between Simple Recovery and Full Recovery for the Microsoft ® SQL Server Recovery Model option?

If you choose the Simple recovery model, you can restore only full or differential backups. You cannot restore backups to any specific point in time. SQL Server truncates the transaction logs when the database reaches a transaction checkpoint, and it leaves no log entries for disaster recovery. You lose any modifications to the data added between the last full or differential backup and the failure.

e) Processor

As a minimum requirement, the server must have an Intel® or AMD processor that supports *Streaming SIMD Extensions 2 (SSE2)*. The recommendation is to have a processor with at least two logical CPUs.

It is also a best practice to use processors with higher core density, such as “Quad” processors, because these processors handle SQL workload growth better. Such growth occurs over time as the database size increases.

For large databases and especially for databases with tens or hundreds of active users, the recommendation is to maintain an average CPU utilization under 50%. This helps prevent a SQL Server from being affected by SQL CPU performance bottlenecks that occur when CPU utilization approaches 75%. At this point, many PDM client applications become sluggish. On very large vaults, this might lead to other side effects such as database deadlocks and blocking chains.

You should always measure average CPU utilization during working hours. One of the most efficient ways to lower the utilization is to increase the number of logical CPUs. For example, a SQL Server with an average CPU usage of 60% and 4 cores will drop to 30% when you increase the number of cores to 8 cores, and to 15% when you increase to 16 cores.

Increasing the number of cores does not mean that specific PDM operations will run faster. Instead, the increase prevents performance issues caused by a high concurrency and increases in the SQL workload.

There are also other methods to lower CPU usage. For example, you can deactivate some vault customizations. However, you must do this after breaking down the SQL workload by PDM operations, and reviewing the results with the Support team.

i) Minimum logical CPUs based on the number of active users

You should consider the values in the following table only during the implementation phase, and only for production environments where on average, a user creates more than 30 document versions per day.

Active Users Count	Minimum Logical CPUs
Active users < 10	1
10 ≤ Active Users < 50	2
50 ≤ Active Users < 100	4
100 ≤ Active Users	8

After the SOLIDWORKS PDM implementation phase is over, you can adjust the number of logical CPUs as long as the average CPU usage during working hours is less than 50% and the CPU random spikes are under 70%.

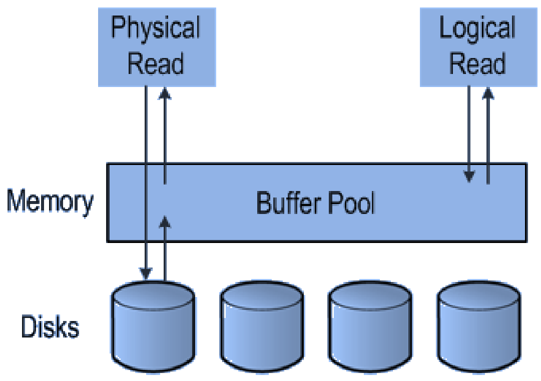
ii) SQL Server editions CPU limitations

SQL Server Platform	Express	Standard	Enterprise
SQL Server 2017	1 Socket or 4 cores	4 sockets or 24 cores	OS Max
SQL Server 2016	1 Socket or 4 cores	4 sockets or 24 cores	OS Max
SQL Server 2014	1 Socket or 4 cores	4 Sockets or 16 cores	OS Max
SQL Server 2012	1 Socket or 4 cores	4 Sockets or 16 cores	OS Max
SQL Server 2008 R2	1 Socket or 4 cores	4 Sockets or 16 cores	8 Sockets

f) Memory

Random Access Memory (RAM) is the key to SQL Server performance; the more RAM, the better (faster) the performance of the database.

SQL Server uses RAM to cache stored procedures, execution plans, and data pages. If there is enough RAM, the buffer pool can store enough compiled plans in caches for later reuse and avoids unnecessary recompiling that may require more CPU.



In a similar way, loading more data pages in the database buffer pool helps reduce and avoid disk I/O. You can accomplish this by increasing memory to use more Logical I/O (which is significantly faster) instead of Physical I/O. If you have more RAM available than the size of your database, the entire database is cached into the buffer pool eventually.

Important:

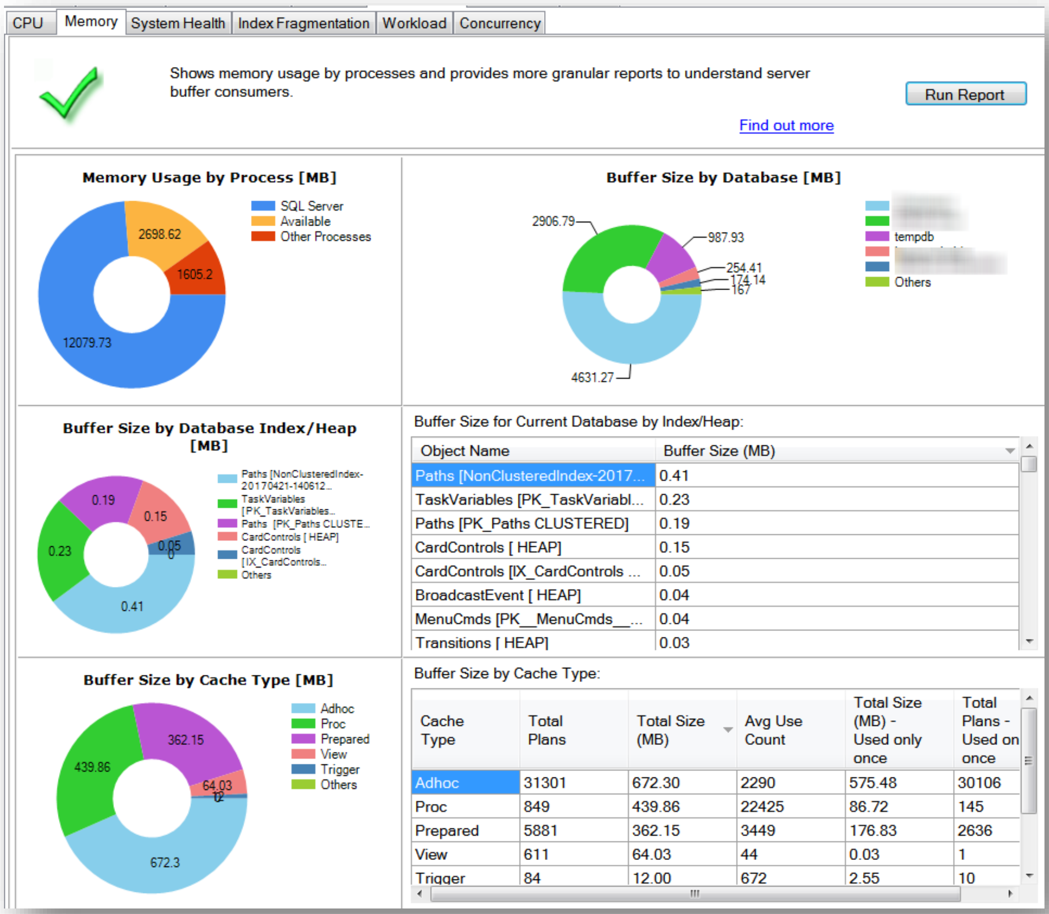
- The minimum requirement for a dedicated SOLIDWORKS PDM server with only one SQL Server instance is 8 gigabytes (GB) of RAM.
- The available RAM should not be less than size of PDM database. If the database size grows over time, add more RAM. Ideally, you want enough memory to contain the entire database plus enough for the Windows OS (perhaps another three GB).
- If additional applications or SQL Server instances share the same server, consider adding more RAM.
- Consider extra capacity for future memory needs.
- When using physical servers, choose higher density memory chips to leave empty slots for future growth.

i) RAM and Databases

Insufficient memory causes SQL Server to read data continually from the disk. Lack of memory significantly affects query performance.

You can check memory usage by using the **PDM Status Report tool** > **Performance** > **Memory** report to understand the memory usage by database, by indexes, and to obtain an estimation of cache type.

This tool is available for Value Added Resellers (VARs) in the SOLIDWORKS Knowledge Base solution *S-066526*. While customers do not have direct access to this tool, VARs can share the tool with customers on case-by-case basis.



Shows memory usage by processes and provides more granular reports to understand server buffer consumers.

Buffer Size for Current Database by Index/Heap:

Object Name	Buffer Size (MB)
Paths [NonClusteredIndex-2017...	0.41
TaskVariables [PK_TaskVariabl...	0.23
Paths [PK_Paths CLUSTERED]	0.19
CardControls [HEAP]	0.15
CardControls [IX_CardControls ...	0.05
BroadcastEvent [HEAP]	0.04
MenuCmds [PK__MenuCmds__...	0.04
Transitions [HEAP]	0.03

Buffer Size by Cache Type:

Cache Type	Total Plans	Total Size (MB)	Avg Use Count	Total Size (MB) - Used only once	Total Plans - Used on once
Adhoc	31301	672.30	2290	575.48	30106
Proc	849	439.86	22425	86.72	145
Prepared	5881	362.15	3449	176.83	2636
View	611	64.03	44	0.03	1
Trigger	84	12.00	672	2.55	10

The first thing to evaluate is if the most important database is using the most RAM. The more RAM the database has allocated from the available RAM (assuming it is a restricted commodity), the faster the performance of the database. If some nonessential databases are using RAM, consider moving them.

ii) Sharing RAM

All databases on the server share the available RAM, and SQL Server manages which databases receive RAM. There is no way to configure one database to use more or less RAM than another database on the same server. There is also no way to specify that one database receives a higher priority than another database. All databases receive equal resources.

A common problem with SQL Server configurations is when Database Administrators (DBAs) run test databases on the same server as the production databases. The DBAs might think that this is not a problem because thousands of users are making thousands of queries against the production databases and only a handful of users are accessing the test databases. They might assume that the test databases will consume resources that are proportional to the number of users and queries that are calling them. This is true for CPU resources. However, this is not true for the shared RAM.

For example, if a tester using a test database runs a query that selects all rows in a table, and those rows are paged from the hard disk into RAM, this might cause the data in the production database to be paged out of RAM and onto the hard disk. A split second later, the thousands of users who are making thousands of requests make their queries, and the production data must reload from the hard disk and back into RAM. In this case, the tester is severely affecting the performance of the production queries even though the ratio of work is 1 to 1,000. You can watch this happen by running such a query while working in the different databases. Take note of the changes in the database sizes.

RAM is a precious resource that directly affects performance. As a best practice, do not share RAM across databases that have different priorities.

iii) Limiting the memory that SQL Server can allocate

By default, SQL Server dynamically consumes memory based on available system resources. It is possible that the SQL Server process ***sqlservr.exe*** will consume all available memory, leaving the OS and other processes short of memory. To control this, the SQL Server **Maximum server memory** option makes it possible to cap the amount of memory that the SQL Server buffer pool can use. The default value for this setting is 2 TB, which is likely never reached.

To activate the **Maximum server memory** option:

1. Open **SQL Server Management Studio** and log in to the SQL Server.
2. Under **Object Explorer**, right-click the SQL Server name > **Properties** > **Memory**.
3. For **Maximum server memory**, specify a value (in MB). For example, to use a maximum of 6 GB RAM, set the value to ***6000***. You can apply the setting without restarting the service.

A similar **Minimum server memory** option is also available. The recommendation is that you use the default value of **0** (zero) for this option.

iv) Memory limitation by SQL Server edition

The following memory limitations apply per SQL Server edition and version.

SQL Platform	Express	Standard	Enterprise
SQL Server 2017	1410 MB	128 GB	OS Max
SQL Server 2016	1410 MB	128 GB	OS Max
SQL Server 2014	1 GB	128 GB	OS Max
SQL Server 2012	1 GB	64 GB	OS Max
SQL Server 2008 R2	1 GB	64 GB	OS Max

g) Disk Configuration

Storage Area Network (SAN) is storage on a remote computer that appears locally attached to the OS. For performance reasons, avoid using SAN as the disk storage for SQL Server, if you share the storage with other applications. For more information about this scenario, see the following Knowledge Base solution:

S-056773 When using a storage area network (SAN) what could cause SOLIDWORKS® PDM performance issues?

Redundant Array of Inexpensive Disks (RAID) storage is locally attached to the computer running the operating system. In a SOLIDWORKS PDM environment, consider using hardware-based RAID. Do not use software-based RAID.

i) Database size limitations for SQL Server editions

SQL Server Platform	Express	Standard	Enterprise
SQL 2017	10 GB	524 PB	524 PB
SQL 2016	10 GB	524 PB	524 PB
SQL 2014	10 GB	524 PB	524 PB
SQL 2012	10 GB	524 PB	524 PB

ii) File architecture

SQL Server databases for SOLIDWORKS PDM use two types of physical files for data storage:

- **Primary data files**

The primary data file is the starting point of the database and points to the other files in the database. Every database has one primary data file. The recommended file name extension for primary data files is **.MDF**. There can also be secondary data files (usually with the extension **.NDF**) which, when combined with the primary MDF file, are known as a *filegroup*.

- ### Log files

Log files hold all of the logging information that system uses to recover the database. There must be at least one log file for each database, although there can be more than one. The recommended file name extension for log files is **.LDF**.

By itself, SQL Server does not enforce the MDF and LDF file name extensions. However, these extensions help identify the different kinds of files and their use.

In a SOLIDWORKS PDM system, you find the following physical files:

Datafile Name	Logfile Name	Description
master.mdf	mastlog.ldf	Standard SQL Server database
model.mdf	modellog.ldf	Standard SQL Server database
MSDBData.mdf	MSDBLog.ldf	Standard SQL Server database
tempdb.mdf	templog.ldf	SQL Server temporary database
ConisioMasterDb.mdf	ConisioMasterDb_log.ldf	SOLIDWORKS PDM master database
<Vault_Name_1>.mdf	<Vault_Name_1>_log.ldf	SOLIDWORKS PDM Vault 1 database
<Vault_Name_2>.mdf	<Vault_Name_2>_log.ldf	SOLIDWORKS PDM Vault 2 database

Creation of each SOLIDWORKS PDM vault generates the initial MDF and LDF files based on the default values. It is these files that you want to work on to optimize performance. The vault-specific data files and the *tempdb database* (tempdb) files (discussed later in this document) are the only files to consider from a performance viewpoint.

iii) Optimizing the database files

If you want to run a big SQL Server database and maintain high performance, begin by sizing up the database files. Pre-sizing the database files appropriately enables the best SOLIDWORKS PDM performance.

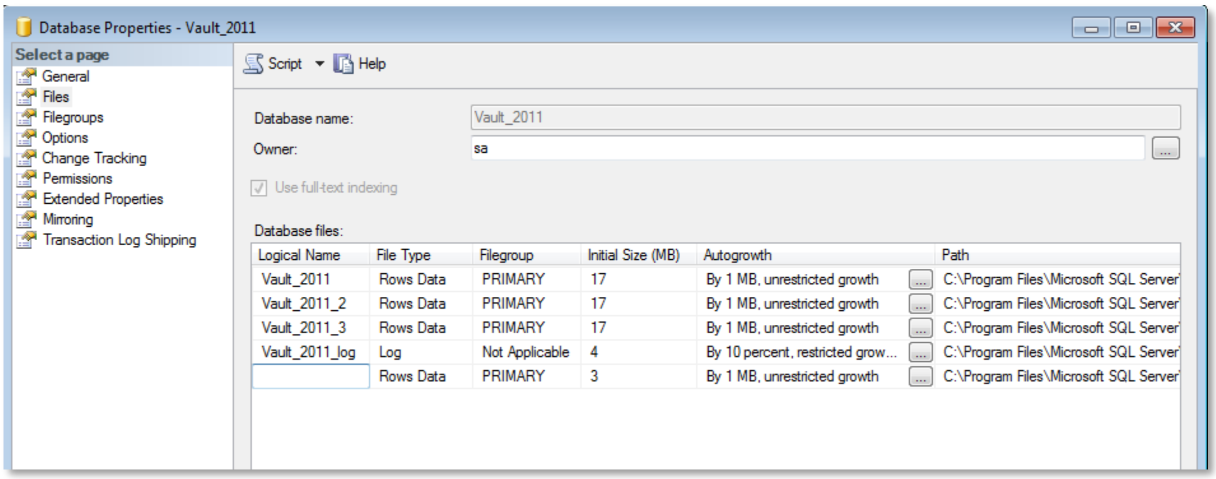
When you create a database for the first time, do the following:

- Create the files in a contiguous space that is defragmented.
- Make all database files the same size.
- Make the database files as large as you anticipate you will ever need.

To defragment the hard disks, you can use a standard operating system disk utility like *Disk Defragmenter*, which is included in the Windows Server operating system. However, be careful if database files already exist on the drive, because the disk defragmentation process moves pieces of your database files around, which causes internal fragmentation of the files. If this happens, rebuild your indexes after defragmenting the drive.

iv) Multiple database files

You might choose to add extra data storage by creating additional data files, rather than by extending the single data file that the installation creates as a default. You can use SQL Server Management Studio to accomplish this.



Database Properties - Vault_2011

Database name: Vault_2011

Owner: sa

☑ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth	Path
Vault_2011	Rows Data	PRIMARY	17	By 1 MB, unrestricted growth	C:\Program Files\Microsoft SQL Server
Vault_2011_2	Rows Data	PRIMARY	17	By 1 MB, unrestricted growth	C:\Program Files\Microsoft SQL Server
Vault_2011_3	Rows Data	PRIMARY	17	By 1 MB, unrestricted growth	C:\Program Files\Microsoft SQL Server
Vault_2011_log	Log	Not Applicable	4	By 10 percent, restricted grow...	C:\Program Files\Microsoft SQL Server
	Rows Data	PRIMARY	3	By 1 MB, unrestricted growth	C:\Program Files\Microsoft SQL Server

Alternatively, you can also use a SQL command similar to the following:

```sql
--Create additional data file
ALTER DATABASE VAULT_2011
ADD FILE (NAME = Vault_2011_2,
FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL10_50.SQLINST\MSSQL\DATA\Vault_2011_1.ndf',
SIZE = 16 MB) ;
GO
```

This approach to creating additional data files within the filegroup only improves performance if you can place the data files on separate physical disks. If this is not possible, then extending and defragmenting the single data file will have the same effect.

If a filegroup contains more than one file, ensure that all files in the filegroup are the same size. Doing this changes the SQL Server extent allocation strategy and creates the entire extent in a single file. SQL Server allocates space sequentially in extents, or units, of 64 KB. It uses a "proportional fill" strategy, which means that SQL Server tries to keep all of your database files at the same percentage of fill.

If all of your database files are the same size, SQL Server allocates 64 KB in the first file, then moves to a second file and allocates 64 KB, and so on until it wraps back around to the first file, where it begins these allocations again. This type of allocation provides the best overall performance. However, if you have files that differ in size, SQL Server tries to maintain the same percentage of empty space in each file by allocating extra extents to the larger files. This uneven file allocation may put all the most recent data in one large file, causing a "hot spot" for the retrieval of that data. This is not ideal. A logical consequence of this point is that the **Autogrow** function is not helpful here. If one file grows enough to initiate an autogrow, the file will no longer be the same size as the other files in the filegroup. To avoid this situation, monitor the capacity of each of the files, and when necessary, manually extend ALL of the files in the filegroup.

v) Understanding RAID for SQL Server

When designing your database server, choosing the right number of hard drives and the correct RAID configuration can save you a lot of time. If you make a mistake, changing the RAID configuration and moving the database later to correct problems on a deployed server will cause long downtimes and consume IT resources.

Before discussion about how to distribute files across disk volumes, here is a brief description of the different types of RAID configurations:

vi) Software RAID

Some operating systems provide a software-based emulation of RAID behavior. As a best practice, do not use software-based RAID as a substitute for an actual hardware RAID implementation to support SQL Server. Software-based RAID provides no performance improvement and is likely to cause additional load on the CPU.

vii) Hardware RAID

RAID is the simulation of a single disk over more than one physical disk drive. There are different methods of performing this simulation, called “levels.” Each RAID level has advantages and drawbacks. This section discusses how each level relates to SQL Server.

• RAID Level 0

RAID level 0, also known as *disk striping*, divides the data into blocks and spreads the data in a fixed order across all disks in an array. RAID 0 improves read and write performance by spreading operations across multiple disks.

Another way to think about RAID 0 is that a single write happens on only one disk, and reads can occur asynchronously across all physical disk heads in the array.

Because your data is valuable, remember that RAID does not always mean “data protection.” RAID level 0 does not provide redundancy and does not protect against disk failure. Do not rely on RAID 0.

• RAID Level 1

RAID level 1 is known as *disk mirroring* because it uses a disk file system called a *mirror set*. Disk mirroring provides a redundant, identical copy of a select disk. All data written to the primary disk also writes to the mirror disk. RAID 1 provides fault tolerance and generally improves read performance although it may degrade write performance. There is a small performance hit at write time to write to two disks. However, there is an advantage at read time when there are two possible sources of the data, so results return more quickly. For a SOLIDWORKS PDM database, which reads more than it writes, this is a significant advantage.

RAID 1 is very suited to SQL Server. It is fast and provides data protection.

However, there are some disadvantages to using RAID 1. It uses two hard disks of identical sizes, which can have several drawbacks. The first drawback is that the size of the logical disk is the same as the size as one of the physical disks. In other words, you pay for two disks, and can only use the storage size of one.

Another more important drawback is that you can only store files of up to the size of one of the disks on the logical disk. For example, if you have two 185 GB drives in a RAID 1 configuration, you end up with a single logical drive of 185 GB. In this example, the maximum file size that you can store on the disk is less than 185 GB. Because your largest database files are the MDF file (which holds the data) and the LDF file (which holds the transaction logs), you need to make sure that these files do not exceed the size of the RAID 1 logical drive.

One way to prevent data from exceeding the size of the RAID 1 drive is to create a secondary (NDF) database file of an equal size on another RAID 1 logical drive set. This divides the data in the database between the two files.

You can prevent the transaction (LDF) log file from becoming too big by making frequent backups of the transaction log. This makes it possible to store the log on a reasonably sized disk. The backup file for the transaction log file has the file extension **.TRN**.

- **RAID Level 5**

RAID level 5 is also known as *striping with parity*. The parity information provides data redundancy. RAID 5 arranges the data and parity information on the disk array so that the two types of information are always on different disks. A RAID 5 configuration allows more than two physical drives in the RAID configuration. In fact, with RAID 5, there is no benefit until you have three or more drives. The data writes to all drives at the same time. However, data reads occur from any of the drives without involving the other physical drives in the array.

The amount of storage on the logical drives is equal to the sum of all available physical disk storage minus one disk. For example, if you have three 185 GB drives, you have 370 GB of logical storage (3 x 185 – 185). With four 185 GB disks, you have 555 GB of logical storage (4 x 185 – 185). This makes it possible to design any size logical drive to fit even the biggest database file.

The issue with RAID 5 is that adding disks to the array increases the cost of writing data. One poor approach is to assign all of the physical disks on the server into one large RAID 5 drive. The thought behind this approach is that the IT administrator and the DBA would never have to worry about individual drives running out of room, about disproportionate file growth across separate arrays, or about having to move and manage files.

The reason this approach is a bad idea is that each write to the RAID is committed on all disks, no matter what data is written (such as database data files, transaction logs, Windows virtual file, or other data). Therefore, for a computer that is running SQL Server, it is not a good approach to use RAID 5 arrays over a large number of disks.

- **RAID Level 10**

RAID level 10 is also known as *mirroring with striping*. RAID 10 uses a striped array of disks that then mirror to another identical set of striped disks. For example, consider creation of a striped array of five disks. This striped array of disks is then mirrored by using another set of five striped disks. RAID 10 provides the performance benefits of disk striping with the disk redundancy of mirroring. RAID 10 also provides the highest read-and-write performance of any RAID level, at the expense of using twice as many disks.

RAID 10 represents the best of both worlds. It offers fast write times and infinitely large logical drives. The extent of your ability to deploy RAID 10 is dependent on the drives that you have. The minimum number of drives for RAID 10 is four. In some cases, having two sets of RAID 1 drives perform better than dedicating all four of those drives to a single RAID 10 array.

For RAID 1, you need at least two physical drives; for RAID 5, you need at least three drives; and for RAID 10, you need at least four drives. The optimal RAID level for your SQL Server file subsystem depends on the number of drives that you have available and the data that you want to store. The next section discusses how the count of physical disks affects your SQL Server configuration and your RAID choices.

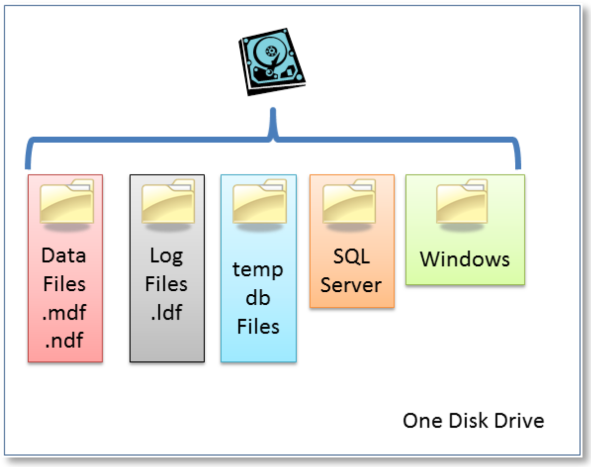
viii) How many hard drives do I need to support SQL Server?

The most appropriate RAID configuration and best method to distribute the database files depends on the number of hard drives available in your server. The number of hard drives you want to deploy is probably a matter of economics as well as performance and robustness. There is no simple “best” arrangement because there must be an element of compromise between the conflicting factors.

For perspective, the following scenarios illustrate how to best utilize the available hardware resources.

- **One drive**

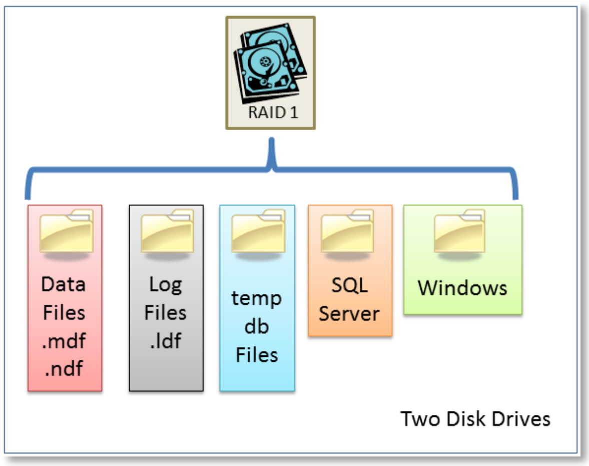
If you have a single drive, you will experience major performance issues. A single drive does not provide redundancy of your file system, and you cannot use RAID. SQL Server runs on a single disk, but doing so compromises both performance and robustness.



If you must use a single disk drive, make sure that you use the fastest drive available (the shortest seek time) and that you defragment the drive regularly.

- **Two drives**

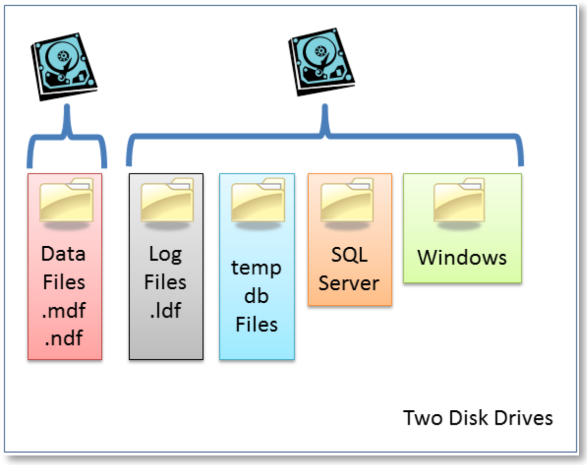
With a two-drive SQL Server installation, you can deploy RAID 1 (disk mirroring). In this scenario, all data (Windows, SQL Server, database data files, transaction logs, and tempdb files**)** goes onto the one logical drive. This deployment is fast and redundant; however, it creates a bottleneck around the disk I/O.



When it comes to hardware, the disk subsystem has the potential to become the greatest bottleneck to SQL Server.

The speed of the disk drives and the ability of the drives to read and write data are the main factors that cause a server running SQL Server to experience slow performance. In other words, the more disk heads you have, the faster the server will perform. Therefore, while two drives that use RAID 1 create a safe environment, it is best to have as many disk heads (physical disk spindles) in your deployment as possible.

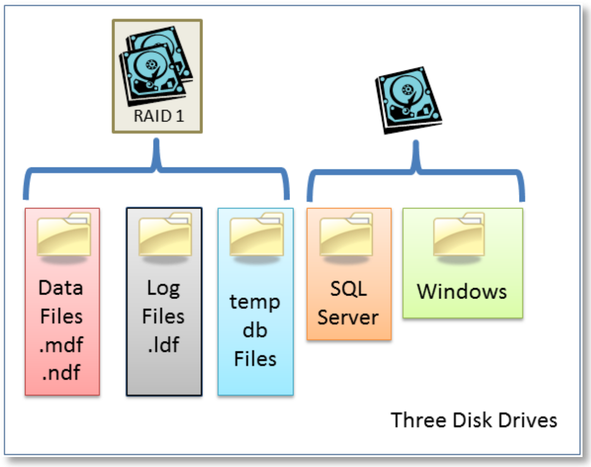
A better alternative is to distribute the load across the two disks without using RAID. This method trades robustness for performance. In this case, allocate the one disk drive to the database data (MDF) files. Everything else including the operating system, the tempdb files, the virtual paging file (an operating system file), and the transaction log go onto the primary drive.



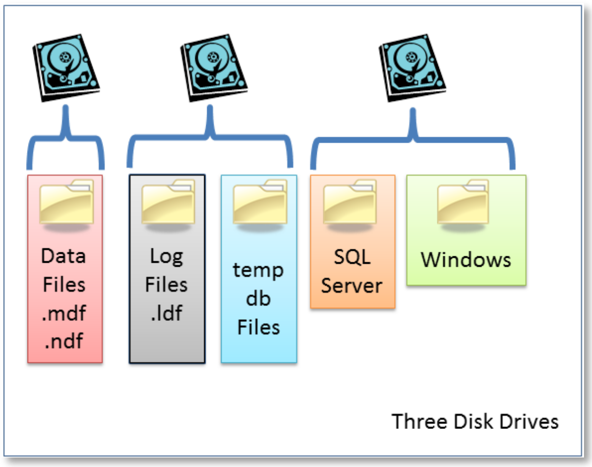
- **Three drives**

With three drives, you can deploy RAID 5, which provides better performance for reading data. However, the drawback is a decrease in performance when writing data. RAID 5 also uses more physical disk space than with a two-drive deployment.

A more likely scenario is to gain the read and security benefits by using a RAID 1 array for all the database files. This means trading this robustness against the likely disk contention.



If you can manage the risks around reducing robustness, you can achieve better performance by spreading the files across separate disk spindles.

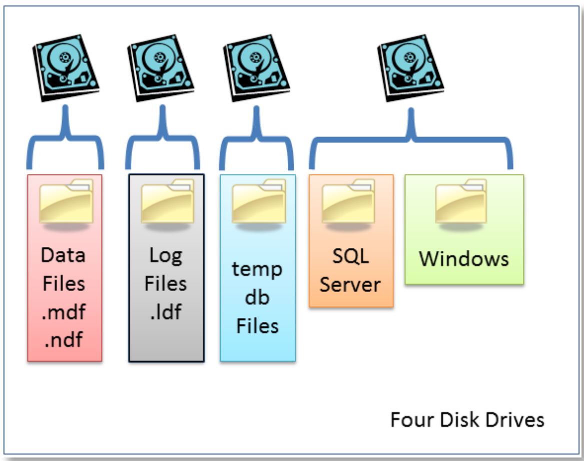


- **Four drives**

With four drives, things get interesting. You could deploy RAID 10 and create one logical drive that holds all data (Windows, SQL Server, the database data files, the transaction log, and the tempdb files). This is very tempting because there is a lot of publicity about the benefits of using RAID 10.

However, it is better to deploy two sets of RAID 1 arrays, which give you two logical drives. This way, you can move the database data for SOLIDWORKS PDM to the second logical drive, and keep the tempdb files, the virtual paging file, and the transaction log on the first logical drive. Typically, the system does not access the virtual paging file very often.

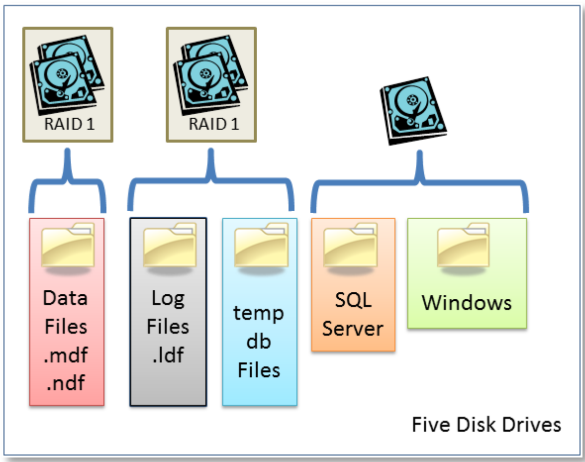
Another option is to not use RAID at all. Instead, you could spread the data over four physical disks. This option maximizes speed, but the system is less robust.



- **Five drives**

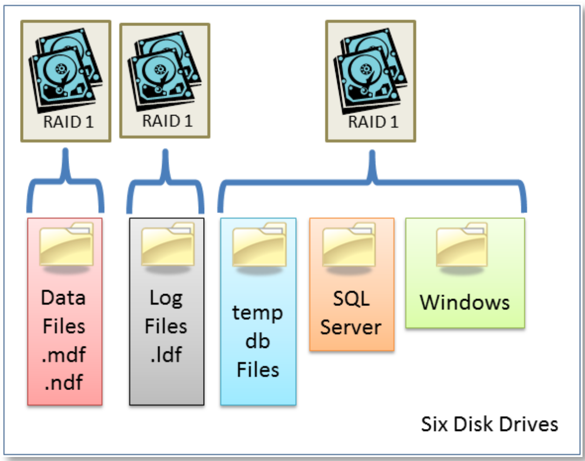
With five drives, you could deploy two of the drives in a RAID 1 array and the other three in a RAID 5 array. This scenario provides two logical drives. Like the four-disk scenario, you want to separate the database for SOLIDWORKS PDM from the rest. Therefore, move the database data (MDF) files to the RAID 5 array, and leave everything else (including the operating system installation) on the RAID 1 array.

Alternatively, you could use four disks to configure two RAID 1 arrays for the main working files. This leaves the remaining drive for the program files and the OS.



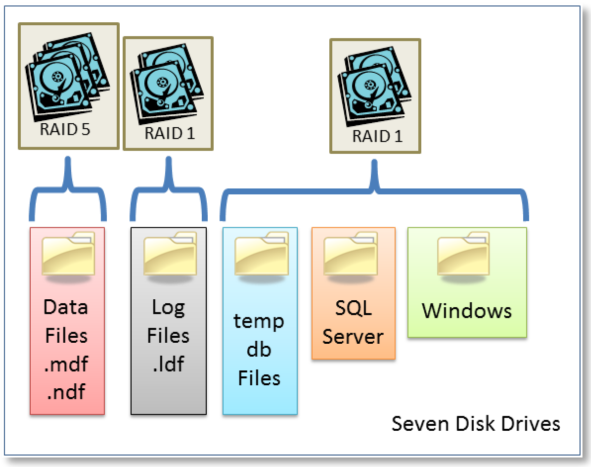
- ## Six drives

The availability of six drives provides many more options for RAID configuration. Six disks perform better than any of the previous scenarios. With six drives in your physical server, you gain the best performance by setting up three arrays of two disks per array in a RAID 1 configuration. Make sure that at least one of the arrays has enough room to hold your MDF file with room to grow. One array should hold the MDF file for SOLIDWORKS PDM and nothing else. One array should hold the transaction log (LDF) files and nothing else. The last array (drive C:) should have everything else, including the operating system installation and the tempdb files.



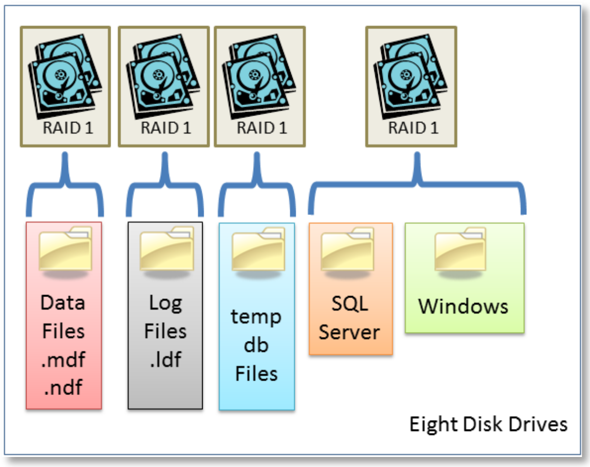
- ## Seven drives

Seven drives allow you to store more data than six drives because you can add the extra drive to one of the arrays, and configure it as a RAID 5 array. Therefore, you still have three arrays, two arrays with two drives in RAID 1, and one array with three drives in RAID 5. On the RAID 5 array, you have the database (MDF) files for SOLIDWORKS PDM. On one of the RAID 1 arrays, you have the transaction log files. The last array (drive C:) should have everything else, including the operating system installation and the tempdb files.



- **Eight drives**

With eight drives, you can create four RAID 1 arrays. You can then put the database files on one array; the transaction logs on another array, the tempdb files on a third array, and everything else on the fourth array (drive C:). This is a very powerful configuration.



- **Summary**

The general best practices for disk setup are:

- Place the database (MDF and NDF) files on their own RAID 1, 5, or 10 arrays or Logical Unit Number (LUN) for SAN.
- Place the transaction log (LDF) files on their own RAID 1 array or LUN.

- Place the tempdb data on its own RAID 1 or 10 arrays or LUN for a large database.
- Place the operating system and SQL Server binaries on a RAID 1 array.

h) Configure Tempdb

Tempdb is an internal built-in database that SQL Server uses to manage temporary objects such as temporary tables, sort tables, and various other objects. Depending on the loading and usage patterns for a given environment, the configuration of tempdb can become a factor in overall performance. The key factors that can improve performance are as follows:

- Move tempdb files from the default location to one or more fault-tolerant disks.
- Where practical, give tempdb dedicated use of disks.
- Configure the startup size of tempdb to minimize autogrow.
- Create multiple data files for tempdb to reduce contention.

i) Move tempdb files

By default, tempdb files are located in the same directory as other system databases for SQL Server. For a busy server, this can affect performance. Therefore, consider moving tempdb files to a high-performance, fault-tolerant disk volume such as RAID 1 or RAID 10 arrays on fast disks. As a best practice, do not place tempdb files on a RAID 5 array, because RAID 5 has inferior write performance.

You use the ‘ALTER DATABASE statement to move tempdb files. For example, to move tempdb files to the **G:** drive and the transaction log files to the **H:** drive, open a SQL Server Management Studio query window and perform the following commands:

```
ALTER DATABASE tempdb
MODIFY FILE (NAME = tempdev, FILENAME = 'G:\SQLData\tempdb.mdf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = templog, FILENAME =
'H:\SQLData\templog.ldf');
GO
```

For this change to take effect, you must restart SQL Server. Note that SQL Server creates fresh copies of tempdb data and log files every time the service starts. Therefore, you do not need to move the actual files manually. The service restart creates files in the new locations that you specify.

ii) Use Multiple, dedicated disks

If your server has sufficient disk resources, follow the typical best practices for database file placement for the tempdb files. The following table summarizes these best practices.

Available for tempdb	Strategy
One disk	Place all tempdb data and log files here, preferably with no other heavily used files on this volume.
Two disks	Place the tempdb log on one volume. Place all tempdb data files on the other volume.
More than two disks	Place the tempdb log alone on one volume. Distribute tempdb data files evenly across the remaining volumes.

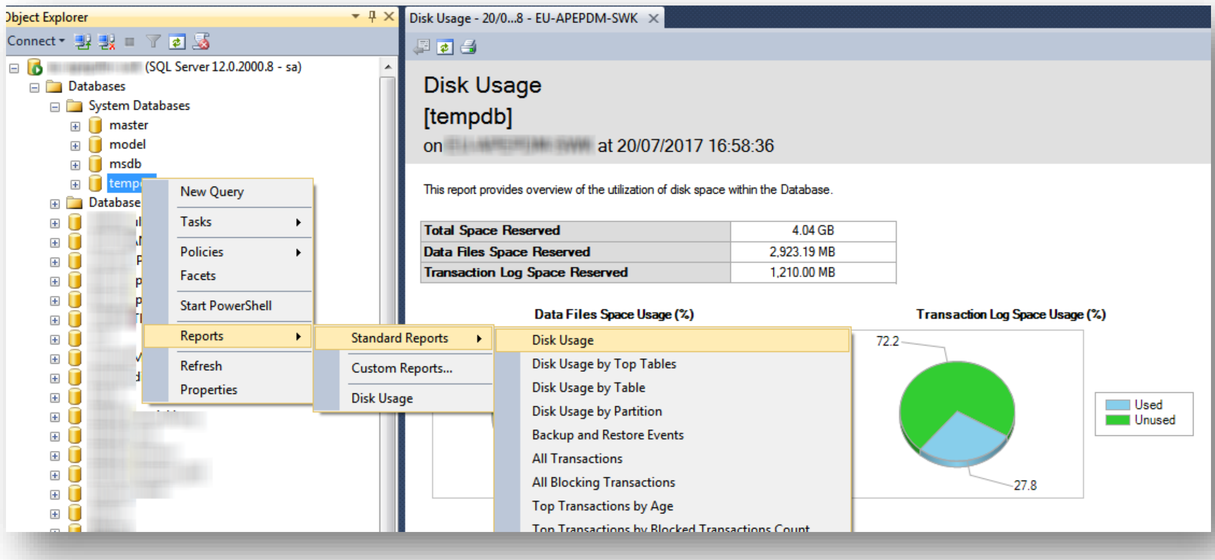
iii) Configure tempdb size

SQL Server uses the tempdb database in a very dynamic way. The server is constantly adding and removing structures within tempdb. At any given time, an operation might cause a (relative) spike in the required size of the database. If tempdb starts out too small for your typical workload, the result may be a large number of frequent autogrowth operations in tempdb. This can have two negative effects. First, when autogrowth occurs, performance can suffer. Second, multiple small autogrowth operations result in heavy fragmentation of the data files. From that time on, there is a degradation of tempdb I/O performance.

To avoid these problems, proactively specify the size of tempdb as appropriate for the environment. There is no fixed rule for the size of tempdb. It depends on the environment.

Two approaches are possible. The first approach is to use an arbitrary rule of thumb and specify a large size for tempdb at the time of installation. For example, specify 2 GB for the database size, and specify 1 GB for the log file size.

The second approach involves observation of the environment in action. This starts with leaving the tempdb sizes at their default level with autogrowth enabled. After a reasonable period of observing the production activity (one or more weeks), the DBA checks the amount of space that tempdb uses. To view the size of tempdb, open SQL Server Management Studio and use the **Standard Reports** option accessible through the **Object Explorer** pane.



Disk Usage
[tempdb]
on [redacted] at 20/07/2017 16:58:36

This report provides overview of the utilization of disk space within the Database.

Total Space Reserved	4.04 GB
Data Files Space Reserved	2,923.19 MB
Transaction Log Space Reserved	1,210.00 MB

The DBA can use the size to which the tempdb data and log files have grown as a good indicator of the target size. Adding a 20% buffer can help ensure minimization of autogrowth.

With this target size in mind, use the `ALTER DATABASE` command to reset the size of tempdb. The following example command sets the database size at 12 GB and the log file size at 10 GB.

```
ALTER DATABASE tempdb
MODIFY FILE (NAME = 'tempdev', SIZE = 12288);
ALTER DATABASE tempdb
MODIFY FILE (NAME = 'templog', SIZE = 10240);
```

Finally, whatever size you choose for tempdb, it is good practice to leave autogrowth enabled on the database. It is the intention to avoid autogrowth if possible, but it is wise to have autogrowth available so that tempdb does not run out of space in an emergency.

For more about tempdb, read the article "*TempDB Database*" at the following Microsoft Online Docs website:

https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15

iv) Create multiple data files

By default, the tempdb database has one data (MDF) file and one log (LDF) file.

If you have a multiple-core server and observer contention around tempdb, you have an option to increase the number of files in tempdb. This can reduce contention in SQL Server internal allocation routines that relate to tempdb.

The number of data files that you should allocate in the tempdb database depends on the number of CPUs (logical or physical) present in the server computer. When SQL Server accesses a database, it starts a scheduler thread for each data file present. Therefore, if you have two CPUs in your physical server, the tempdb database should have two data files to “load balance” the threads.

```
ALTER DATABASE [tempdb]
ADD FILE ( NAME = 'tempdev1',
FILENAME = 'f:\tempdb1.ndf' ,
SIZE = 8192KB , FILEGROWTH = 10%)
ADD FILE ( NAME = 'tempdev2',
FILENAME = 'f:\tempdb2.ndf' ,
SIZE = 8192KB , FILEGROWTH = 10%)
```

There are essentially five points to consider when doing this:

- Strive for one tempdb data file per processor core that is available to SQL Server.
- Maintain an equal size for the data files. SQL Server uses a proportional fill algorithm. Equal size files allow this algorithm to run most efficiently.
- To maintain data files at equal sizes, ensure that the data files have the same autogrowth settings.
- The data files do not need to be on separate physical disks. This recommendation does not depend on splitting I/O across disk devices, but instead exploits the internal I/O and allocation algorithms of the database engine. This differs from the usual guidance for user database files.
- There is no need to create multiple log files.

v) Summary

To summarize the options, consider the following example:

The disk configuration for a server allows you to dedicate two RAID 1 arrays to tempdb (drives G: and K:). The SQL Server software installation is on the C: drive. The server has four processor cores, and all are in use by SQL Server. You decide to set the tempdb size to accommodate a total of 10 GB for data and 5 GB for the transaction logs.

With this in mind, you would run the following T-SQL script in SQL Server Management Studio:

```
--Move and size the tempdb log
ALTER DATABASE tempdb
MODIFY FILE (NAME = templog, FILENAME = 'K:\SQLData\templog.ldf',
SIZE = 5120MB);
GO
--Move and size the default data files
ALTER DATABASE tempdb
MODIFY FILE (NAME = tempdev, FILENAME = 'G:\SQLData\tempdb.mdf',
SIZE = 2560MB, FILEGROWTH = 10%);
GO
--Create additional data files
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev2, FILENAME = 'G:\SQLData\tempdev2.ndf',
SIZE = 2560MB, FILEGROWTH = 10%);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev3, FILENAME = 'G:\SQLData\tempdev3.ndf',
SIZE = 2560MB, FILEGROWTH = 10%);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev4, FILENAME = 'G:\SQLData\tempdev4.ndf',
SIZE = 2560MB, FILEGROWTH = 10%);
GO
```

After running this T-SQL script, you must restart SQL Server for the changes to take effect.

i) Database Replication

Database replication requires the SQL Server Enterprise edition, and uses a primary instance of SQL Server for read and write operations. Remote sites will have at least one secondary SQL Server instance for read-only operations.

There are specific requirements when using database replication. For example, always set the recovery model to **Full**, and use special maintenance indexes that avoid locking database resources, etc.

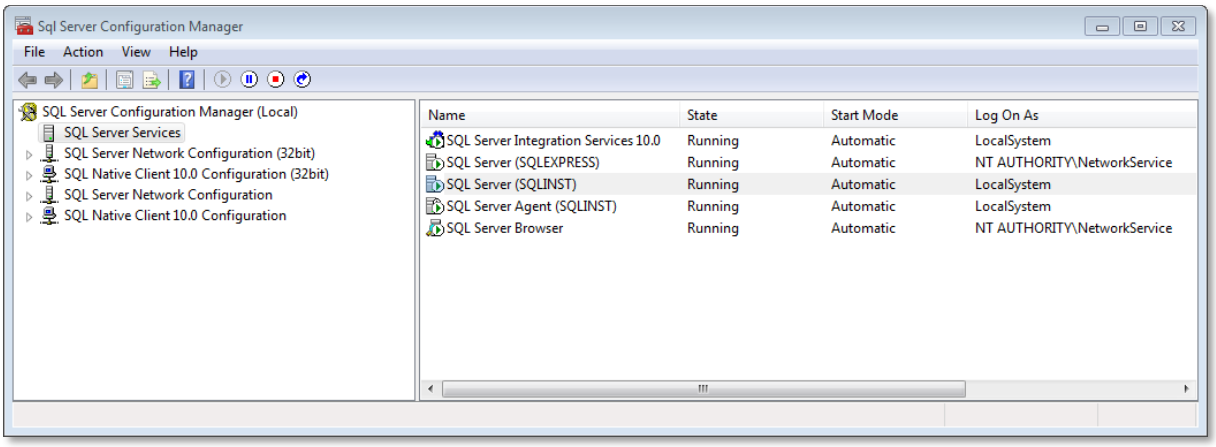
For detailed information about configuring database replication, see the Knowledge Base solution S-072960 *"Using SOLIDWORKS PDM 2017, what are the recommended steps to set up and configure SQL Server Enterprise edition for database replication?"*

j) Configure SQL Services

The following recommendations apply primarily for the SQL Server Standard and Express editions.

- Although your planning phase should determine what SQL services you need to install, there might be occasions when you roll out a service that is not a current requirement. Perhaps your business requires you to install a SQL Server, and although the reporting functionality is not necessary, management insists that you install the **Reporting Services** – “just in case”. To improve performance in cases such as this, disable services that you do not use.

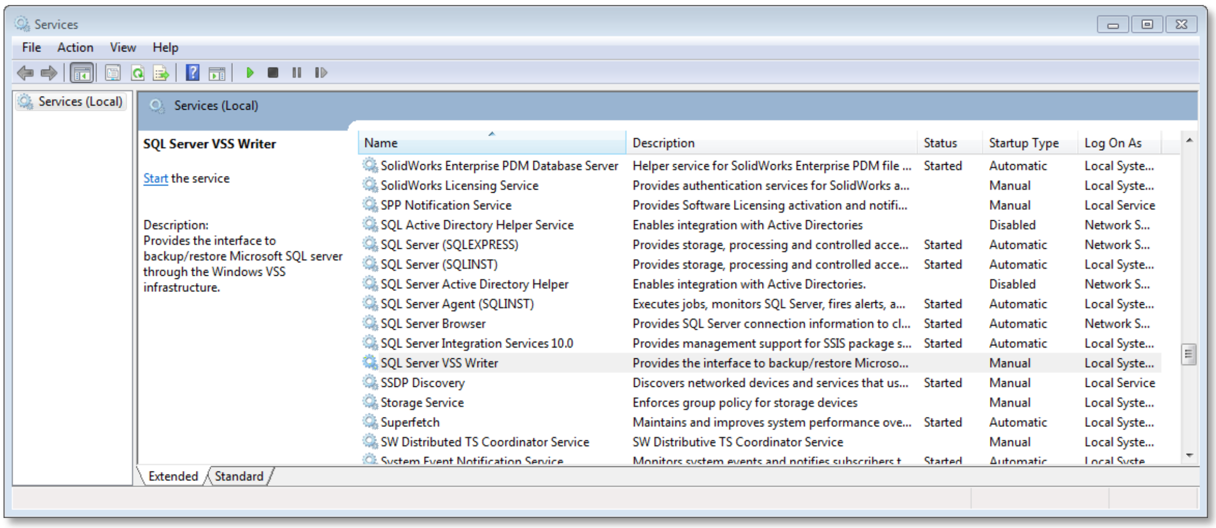
- One example of a service that you can disable is the **SQL Server Browser** service, which SQL Server installs by default. If you are not running named SQL instances, or if you plan to install multiple instances on one computer, you can use the SQL Server Configuration Manager to safely disable this service as depicted in the following image:



Sql Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 10.0 Configuration (32bit)
SQL Server Network Configuration
SQL Native Client 10.0 Configuration

Name	State	Start Mode	Log On As
SQL Server Integration Services 10.0	Running	Automatic	LocalSystem
SQL Server (SQLEXPRESS)	Running	Automatic	NT AUTHORITY\NetworkService
SQL Server (SQLINST)	Running	Automatic	LocalSystem
SQL Server Agent (SQLINST)	Running	Automatic	LocalSystem
SQL Server Browser	Running	Automatic	NT AUTHORITY\NetworkService

Another service that you can disable is the **SQL Server VSS Writer** service. Unless you are using applications that make use of the Windows **Volume Shadow Copy** infrastructure to back up SQL Server databases, you can disable this service safely. Be aware that the **SQL Server VSS Writer** service is not available from the SQL Server Configuration Manager. To access this service, open the Windows **Services** app as shown in the following image.

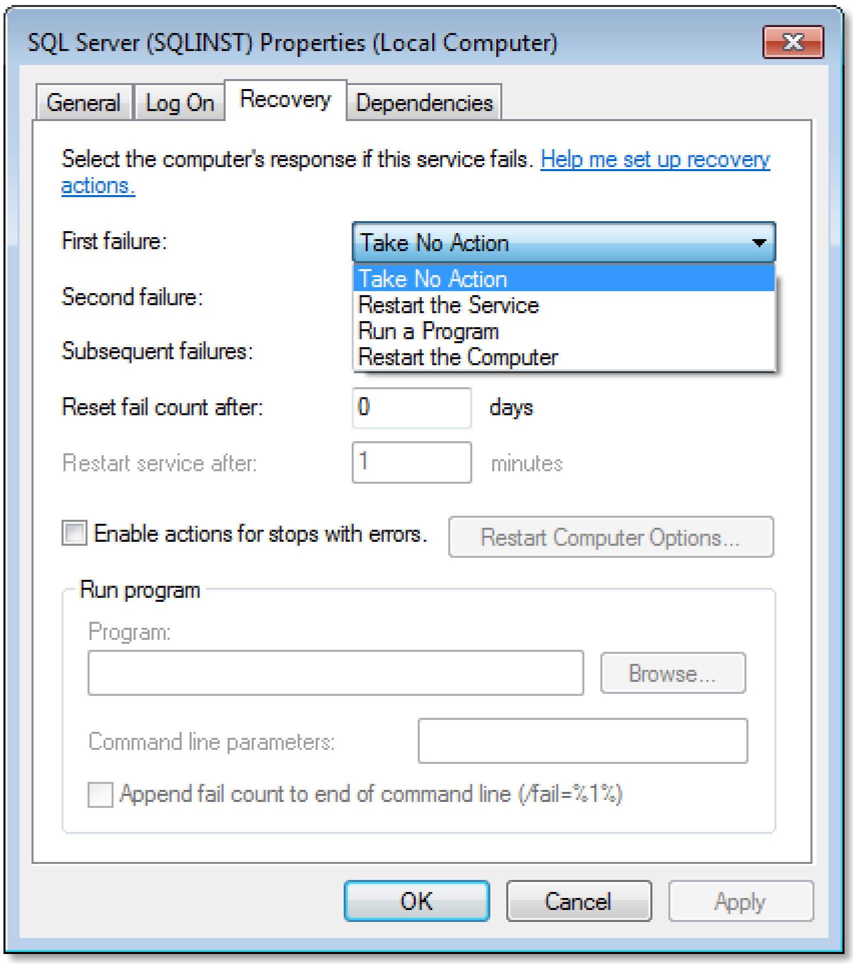


Description:
Provides the interface to backup/restore Microsoft SQL server through the Windows VSS infrastructure.

Name	Description	Status	Startup Type	Log On As
SolidWorks Enterprise PDM Database Server	Helper service for SolidWorks Enterprise PDM file ...	Started	Automatic	Local Syste...
SolidWorks Licensing Service	Provides authentication services for SolidWorks a...		Manual	Local Syste...
SPP Notification Service	Provides Software Licensing activation and notifi...		Manual	Local Service
SQL Active Directory Helper Service	Enables integration with Active Directories		Disabled	Network S...
SQL Server (SQLEXPRESS)	Provides storage, processing and controlled acce...	Started	Automatic	Network S...
SQL Server (SQLINST)	Provides storage, processing and controlled acce...	Started	Automatic	Local Syste...
SQL Server Active Directory Helper	Enables integration with Active Directories.		Disabled	Network S...
SQL Server Agent (SQLINST)	Executes jobs, monitors SQL Server, fires alerts, a...	Started	Automatic	Local Syste...
SQL Server Browser	Provides SQL Server connection information to cl...	Started	Automatic	Network S...
SQL Server Integration Services 10.0	Provides management support for SSIS package s...	Started	Automatic	Local Syste...
SQL Server VSS Writer	Provides the interface to backup/restore Microso...		Manual	Local Syste...
SSDP Discovery	Discovers networked devices and services that us...	Started	Manual	Local Service
Storage Service	Enforces group policy for storage devices		Manual	Local Syste...
Superfetch	Maintains and improves system performance ove...	Started	Automatic	Local Syste...
SW Distributed TS Coordinator Service	SW Distributive TS Coordinator Service		Manual	Local Syste...
System Event Notification Service	Monitors system events and notifies subscribers t	Started	Automatic	Local Syste

- While you should disable non-essential services, other services like the **SQL Server Agent** service are designed to run continuously. This is because scheduled backups, maintenance plans, replication, and jobs are dependent on the **SQL Server Agent** service. Similarly, if your server is will run SQL Service Integration Service (SSIS) packages, make sure that the **SQL Service Integration Services** service starts automatically every time the server starts. You can configure the startup property of a service from the SQL Server Configuration Manager.

- Another thing that is not available in the SQL Server Configuration Manager is the recovery behavior of SQL Services. For example, what should happen if services fail unexpectedly? To manage this behavior, access the service **Properties** from the **Services** app or from the control panel. The **Recovery** tab of the service **Properties** page allows you to define actions for the computer to take if the service fails once, twice, and for subsequent failures. From here, you can specify that the service will restart automatically if it fails, as shown in the following image.


SQL Server (SQLINST) Properties (Local Computer)

General | Log On | Recovery | Dependencies

Select the computer's response if this service fails. Help me set up recovery actions.

First failure: Take No Action
- Take No Action
- Restart the Service
- Run a Program
- Restart the Computer

Second failure:

Subsequent failures:

Reset fail count after: 0 days

Restart service after: 1 minutes

☐ Enable actions for stops with errors. Restart Computer Options...

Run program

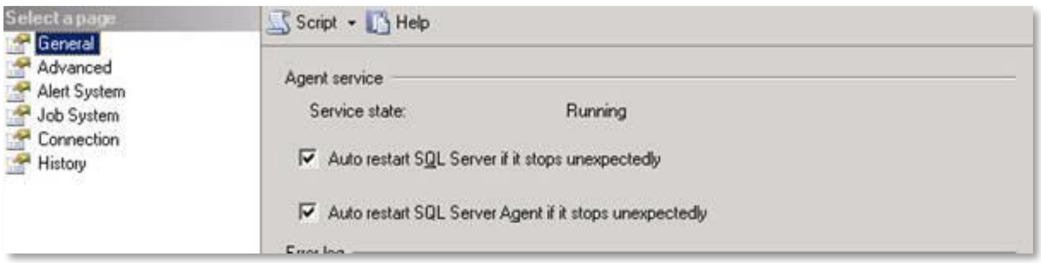
Program: Browse...

Command line parameters:

☐ Append fail count to end of command line (/fail=%1%)

OK Cancel Apply

You can also specify the same type of behavior on the **Properties** page for the SQL Server Agent service. In this case, you can specify that the **SQL Server** service or the SQL Server Agent service restart automatically if they stop unexpectedly.


Select a page
- General
- Advanced
- Alert System
- Job System
- Connection
- History

Script Help

Agent service

Service state: Running

☑ Auto restart SQL Server if it stops unexpectedly

☑ Auto restart SQL Server Agent if it stops unexpectedly

Error log

k) Configure Error Logs

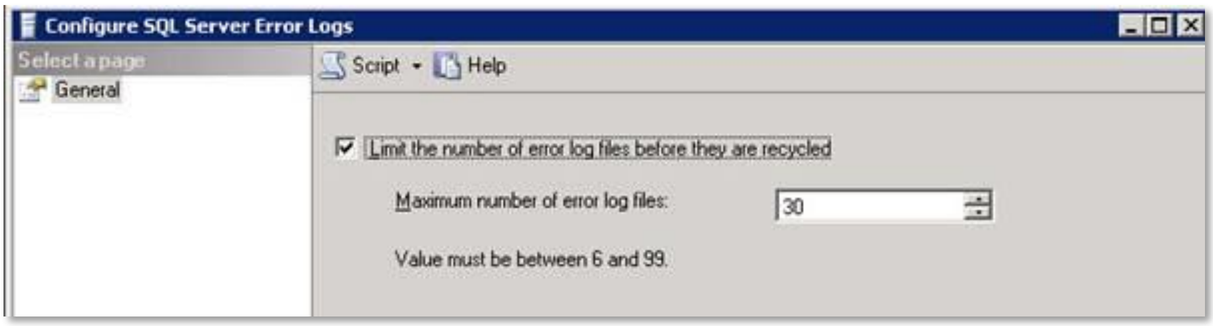
The SQL Server error log is the first place that DBAs look when they troubleshoot server related issues. By default, the server maintains six logs as archive and one log as current. Every time the SQL Server service starts, the system begins a new log file. The name of the log file that is currently active is **ERRORLOG**. The log file before restart (the previous ERRORLOG) becomes archived as **ERRORLOG.1**, the log file before that (with the name ERRORLOG.2) becomes **ERRORLOG.3** and so on.

An annoying thing about the default error log behavior is that any one file can contain a number of days of entries. When you try to view a file, it can take some time to load. If you are interested in the events of a particular day, you need to browse through the entries or use a filter.

In addition, log files are overwritten in a rolling fashion after six files are created. That means that you there is no history earlier than the oldest log file (**ERRORLOG.6**).

To mitigate the default behavior, you can configure SQL Server to maintain a predefined number of log files. You can also create a scheduled job to “recycle” the error logs regularly. Recycling means that SQL Server will start a new log file without restarting the service.

To keep a predefined number of archived logs, use the **Configure SQL Server Error Logs** feature in SQL Server Management Studio. In the following image, the server is set to keep the last thirty log files.

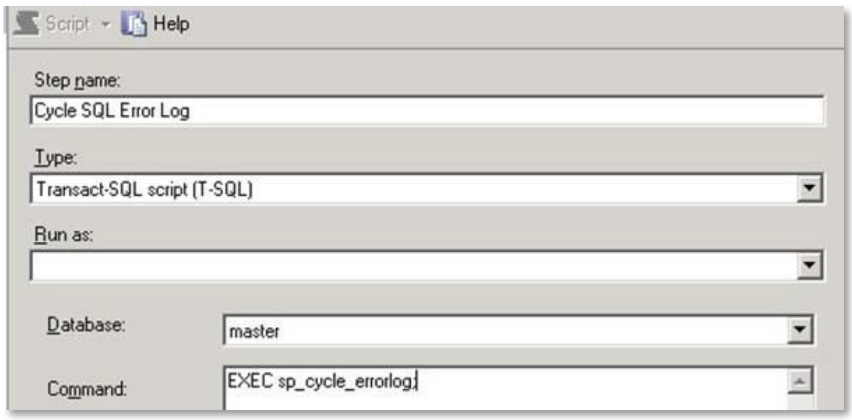


☑ Limit the number of error log files before they are recycled

Maximum number of error log files: 30

Value must be between 6 and 99.

After configuring this property, you can create a scheduled job to run every day to reinitialize the log. The job will have a single step that calls the **sp_cycle_errorlog** system stored procedure, as shown in the following image. This procedure starts a new error log file without restarting the SQL Server service. You could schedule the job to run at 12:00 AM every morning.



Step name: Cycle SQL Error Log

Type: Transact-SQL script (T-SQL)

Run as:

Database: master

Command:

```
EXEC sp_cycle_errorlog;
```

The combined effect of these two changes is that the system will create one new log file every day at midnight. If your server does not restart in between, you will have the last one month of log entries on the disk with each day's log in a separate file.

You can define the same kind of job for the SQL Server Agent error log files. The latest SQL Server Agent error log has the name **SQLAGENT.OUT**. Running the SQL Server **sp_cycle_agent_errorlog** procedure from a job creates a new SQL Server Agent log file and archives the previous file. However, unlike the SQL Server error logs, there is no user interface that allows you to specify the number of SQL Server Agent error logs to keep in the archive.

i) Regular maintenance activities

For the SQL Server Standard and Enterprise editions, you can automate many of the regular maintenance activities by using SQL Server tools called **Maintenance Plans**.

The SQL Server Express edition does provide any maintenance options. To work around this, you can use SQL Server Management Studio Express to create scripts, and then use the Windows Scheduler to run the scripts as tasks. For such an example, see the Knowledge Base solution *S-072048*.

ii) Overview of SQL Server maintenance plans

The DBA can create a database maintenance plan either by using the *Maintenance Plan Wizard* or by using the *SQL Server Integration Services (SSIS) designer*. Using the Maintenance Plan Wizard, you can create basic maintenance plans for all system and user databases. However, creating an enhanced workflow might require that you use the SSIS designer.

This section outlines the different types of maintenance plan tasks that are appropriate for a SOLIDWORKS PDM environment.

To create and manage maintenance plans, you must be a member of the **sysadmin** fixed server role because maintenance plans are only visible to users who are members of this role.

- **Back Up Database Task** - This task allows you to perform different types of database backups such as **Full**, **Differential** or **Transactional Log**, which are based on the recovery model of the **System** or **User** databases.
- **Check Database Integrity Task** – You can use this task to check the allocation and structural integrity of all user and system tables within a database. This task also has an option to check the allocation and structural integrity of all indexes available within a database. Using this task, you can choose the database against which you want to perform the database integrity checks. This task internally runs the ***DBCC CHECKDB*** statement.
- **Execute SQL Server Agent Job Task** – You can use this task to run a SQL Server Agent job that you create on the SQL Server Instance. This task is only available when you use the SSIS designer to create maintenance plans.
- **Execute T-SQL Statement Task** – You can use this task to run Transact SQL queries against a database. This task is only available when you use the SSIS designer to create maintenance plans.
- **History Cleanup Task** – This task deletes historical data that relates to database backups and restore activities, SQL Server Agent job history, database maintenance plan history, etc. This task uses the ***sp_delete_backuphistory*** system stored procedure to clean up the history prior to the number of days, weeks, or months from the current system date.
- **Maintenance Cleanup Task –** You can use this task to remove older files such as maintenance plan execution reports, database backup files, etc. Use this task when you create maintenance plans because the task removes old files that are no longer required.
- **Notify Operator Task** – You can use this task to send messages to the SQL Server Agent Operator when a task completes successfully or fails. The operator can receive notification by email, pager, or by the ***net send*** method. There is more discussion of this in the "*Set up performance alerts*" section of this document.
- **Reorganize Index Task** – Use this task to defragment and compress clustered and non-clustered indexes on tables and views. Reorganization of these indexes is best suited when the indexes are not heavily fragmented. This process takes less system resources than rebuilding an index. If the indexes are heavily fragmented then the best choice is to rebuild indexes.
- **Rebuild Index Task** – You can use this task to organize information on the data and index pages by rebuilding the indexes. This helps to improve the performance of index seeks and index scans. This task also optimizes the distribution of data and free space on the index pages, thereby allowing faster future growth. If you use this task to rebuild the indexes in a single database, then you can choose the views and tables for which you want to rebuild the index. This task also has options such as **Sort results in tempdb** and **Keep index online while re-indexing**. However, these operations require sufficient disk space in the tempdb database because the index is effectively duplicated for the duration of the operation.
- **Shrink Database Task** - You can use this task to reduce the disk space that the database and log files consume by removing the empty data and log pages. When

using this task, the space freed up by shrinking the database can be returned to the operating system, or it can be retained within the database for future growth.

- **Update Statistics Task** - This task ensures that the query optimizer has up-to-date information about the distribution of data values within the tables. This information makes it possible for the optimizer to apply the best strategies for data access strategies.

iii) Daily tasks

- Back up the SQL Server database by using a SQL Server Maintenance Plan.
- Back up Archives by using a specific backup tool or a custom script.
- Back up the SOLIDWORKS PDM environment by using a scheduled task in the Archive Server Configuration tool.
- Move file backups to a secure location, such as away from the SQL Server and Archive server hardware. Do not leave file backups on the source disks or computers.

iv) Weekly and monthly tasks

- Schedule a reorganization and rebuild of indexes by using a SQL Server Maintenance Plan.
- Review the MDF and LDF files to ensure that they do not exceed the initial sizes.
- Check for disk fragmentation.

v) Quarterly tasks

- Check the integrity of the SOLIDWORKS PDM backup strategy, including the SQL Server backup, by restoring and testing a recent system backup in a test environment.

vi) Set up performance alerts

There may be cases where the DBA wants to monitor some system resource in a more proactive way. For example, the DBA might want to review the log files once a month. It is possible to configure SQL Server to monitor various parameters actively, and to take specificed actions such as sending an email notification, when the parameter exceeds the predefined limit.

The process of setting up such an email alert is described in this Solution:

S-027967 *"Is there a way to set up performance email alerts to help monitor a SOLIDWORKS® PDM file vault database in Microsoft® SQL Server?"*

4) Investigating Performance Problems

a) Overview

When poor performance is discovered or reported, there are several areas that you may need to investigate first:

- Check network performance (LAN & WAN)
- Check user authentication
- Check for additional system loading such as new software or new hardware
- Consider reproducing the issue while tracing SQL traffic. If the trace file shows trace events that have a long duration, it means that there is a SQL performance issue. For this task it is recommended to follow the steps in the Knowledge Base solution S-057205.
- Rule out if the issue is SQL related by using a half-splitting testing scenario where you test the same issue both outside and inside a PDM vault.

This document assumes that you have investigated and eliminated those other potential performance problems until only the SQL Server performance possibility remains.

b) Troubleshooting Methodologies

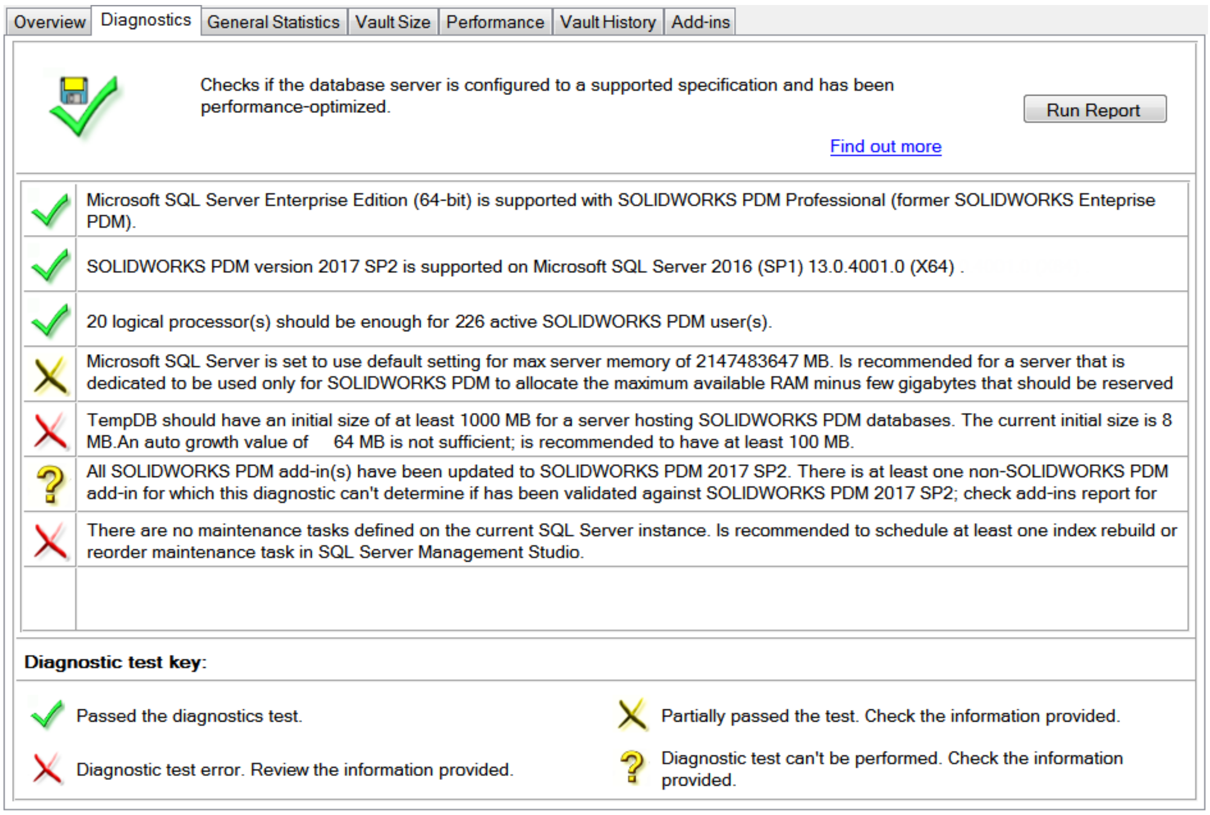
The following troubleshooting process is a guideline that you can use in conjunction with other methodologies or third-party diagnostics and troubleshooting tools.

These methodologies involve troubleshooting tools that are not advisable to use if the SQL Server average CPU utilization is above 90%. For such scenarios, it is best to report the issue to SOLIDWORKS Technical Support and determine if the current methods still apply.

i) Setup check

When troubleshooting SQL Server performance, it is a strong recommendation that you start by confirming that the SQL Server and the SOLIDWORKS PDM vault have the correct configurations. In other words, determine if the configurations adhere to the main guidelines from the "*Installation and Configuration*" chapter.

The best practice to automate the setup check is to involve your local support and use tools such as the *SOLIDWORKS PDM Status Report* tool (available for VARs in the Knowledge Base solution S-066526) that include *Diagnostics* and *Performance* reports that can help identify some of the most common SQL Server performance issues. To use the SOLIDWORKS PDM Status Report tool, connect to a vault and run one or more reports. To access additional technical details for the main reports, click the **Find out more** links within the application.



Checks if the database server is configured to a supported specification and has been performance-optimized.

✓	Microsoft SQL Server Enterprise Edition (64-bit) is supported with SOLIDWORKS PDM Professional (former SOLIDWORKS Enteprise PDM).
✓	SOLIDWORKS PDM version 2017 SP2 is supported on Microsoft SQL Server 2016 (SP1) 13.0.4001.0 (X64) .
✓	20 logical processor(s) should be enough for 226 active SOLIDWORKS PDM user(s).
✗	Microsoft SQL Server is set to use default setting for max server memory of 2147483647 MB. Is recommended for a server that is dedicated to be used only for SOLIDWORKS PDM to allocate the maximum available RAM minus few gigabytes that should be reserved
✗	TempDB should have an initial size of at least 1000 MB for a server hosting SOLIDWORKS PDM databases. The current initial size is 8 MB.An auto growth value of 64 MB is not sufficient; is recommended to have at least 100 MB.
?	All SOLIDWORKS PDM add-in(s) have been updated to SOLIDWORKS PDM 2017 SP2. There is at least one non-SOLIDWORKS PDM add-in for which this diagnostic can't determine if has been validated against SOLIDWORKS PDM 2017 SP2; check add-ins report for
✗	There are no maintenance tasks defined on the current SQL Server instance. Is recommended to schedule at least one index rebuild or reorder maintenance task in SQL Server Management Studio.

Diagnostic test key:

✓	Passed the diagnostics test.	✗	Partially passed the test. Check the information provided.
✗	Diagnostic test error. Review the information provided.	?	Diagnostic test can't be performed. Check the information provided.

ii) Collecting performance data

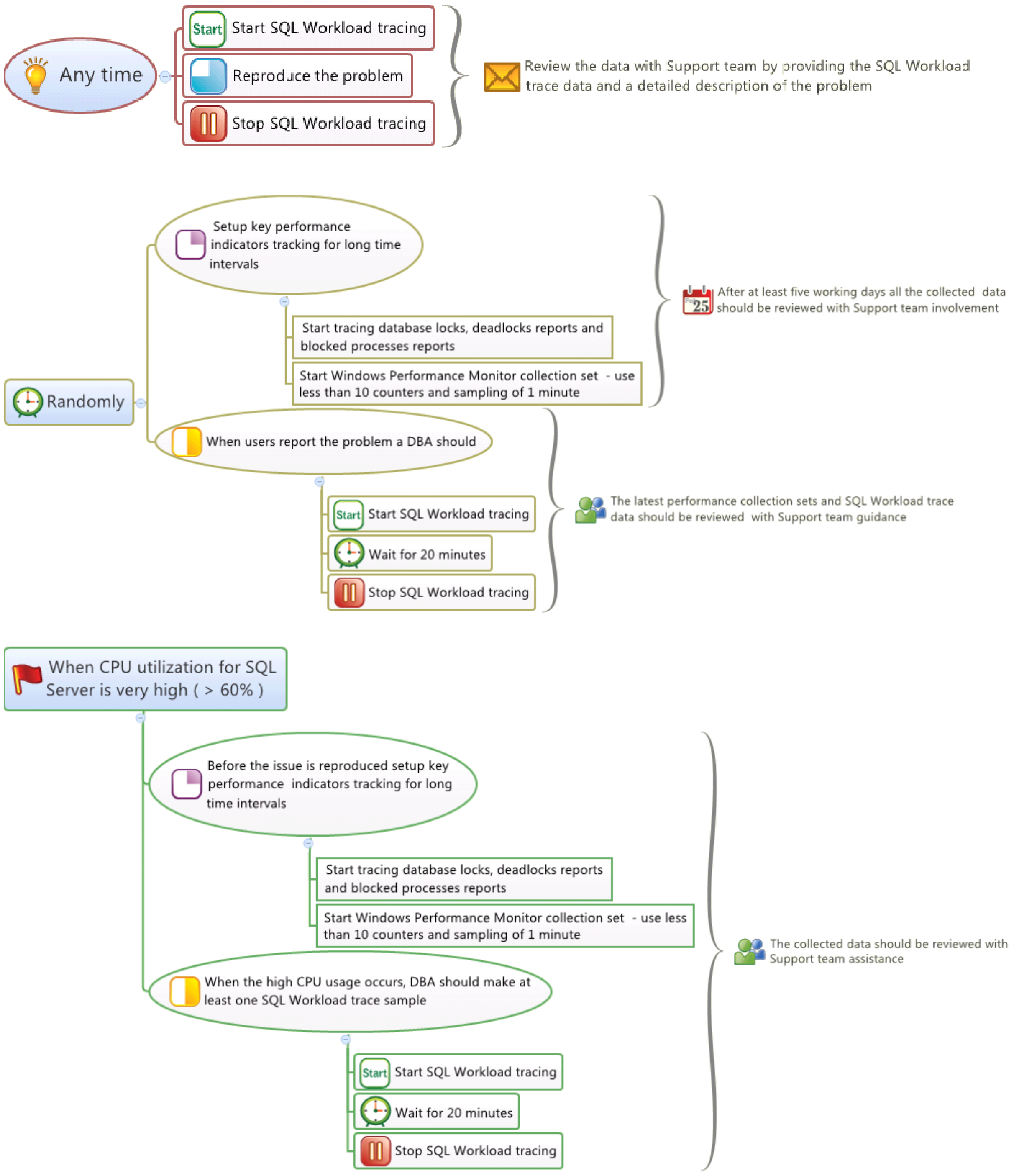
The preferred method of measuring SQL Server performance data is to use Microsoft solutions such as Windows *Performance Monitor*, SQL Server *Extended Events*, or SQL Server traces.

SOLIDWORKS Technical Support recommends the following three methods of data collection. There is more information about these methods later in this document:

- Windows Performance Monitor collection set – It is a strong recommendation to use this collection set with few counters and a one-minute sampling. For more information, see the "*Monitoring performance at all times*" section of this document.
- Tracing random performance indicators such as database locks, deadlocks reports, and blocked processes reports. This method has the advantage that it can run continuously for months. In some cases, it might help identify a performance issue without the need to start a SQL workload trace.
- SQL Workload tracing – This is one of the most detailed collection sets. In some cases, this method can help identify if a specific SOLIDWORKS PDM operation, client computer, application, or file set is the cause of the issue.

The following diagram explains how to use these data collection methods depending on how the issue is reproducible in a production environment.

- **How can you reproduce the problem?**


Any time
Start SQL Workload tracing
Reproduce the problem
Stop SQL Workload tracing
Review the data with Support team by providing the SQL Workload trace data and a detailed description of the problem
Randomly
Setup key performance indicators tracking for long time intervals
Start tracing database locks, deadlocks reports and blocked processes reports
Start Windows Performance Monitor collection set - use less than 10 counters and sampling of 1 minute
After at least five working days all the collected data should be reviewed with Support team involvement
When users report the problem a DBA should
Start SQL Workload tracing
Wait for 20 minutes
Stop SQL Workload tracing
The latest performance collection sets and SQL Workload trace data should be reviewed with Support team guidance
When CPU utilization for SQL Server is very high (> 60%)
Before the issue is reproduced setup key performance indicators tracking for long time intervals
Start tracing database locks, deadlocks reports and blocked processes reports
Start Windows Performance Monitor collection set - use less than 10 counters and sampling of 1 minute
When the high CPU usage occurs, DBA should make at least one SQL Workload trace sample
Start SQL Workload tracing
Wait for 20 minutes
Stop SQL Workload tracing
The collected data should be reviewed with Support team assistance

Important:

- For reference crosschecking of SQL events, it is a best practice to collect and review the following logs with the Support team: SQL Server logs, SOLIDWORKS PDM client and service logs (COG files), and Windows Application logs (EVTX files).

- Random problems or problems that occur when CPU utilization exceeds 60% require SQL workload tracing. A DBA or a SOLIDWORKS PDM administrator that is familiar with SQL Server Extended Events and SQL server side tracings should perform this task.
- Before reporting an issue that is reproducible at any time to Technical Support, confirm that the problem is also reproducible in a test environment.
- For most performance cases, the data mentioned in this section should be enough to understand the main SQL performance issues. However, in some complex cases, Technical Support could ask to adjust the collection sets or to repeat the data collection with a different server or database configuration.
- In cases where the collected data cannot help identify the performance issue, the recommendation is to work directly with end users and review their actual complaint. This may reveal that the actual problem might not relate to only the SQL Server performance.

iii) Analyzing performance data

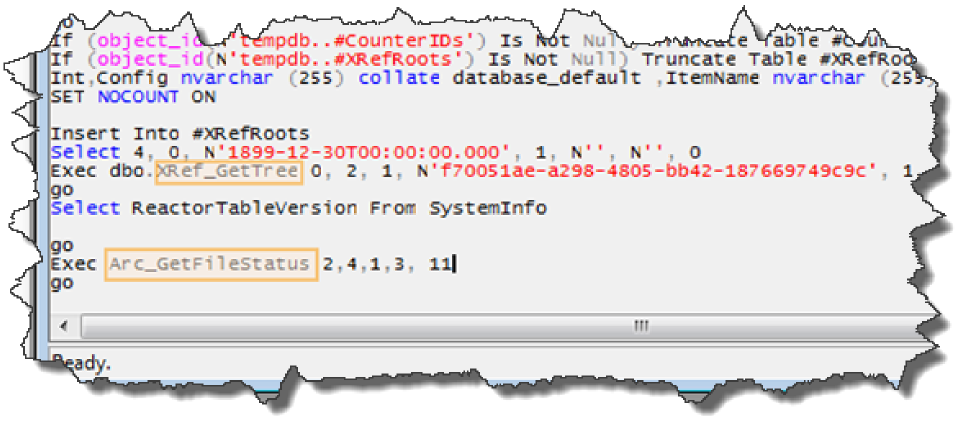
Following the process in the previous section ensures the collection of all main key performance indicators for SQL Server. In addition, SQL trace (TRC file format) or Extended Events sessions (XEL file format) include details with the SQL query execution.

As a general rule, the more deadlocks, blockings reports, and high CPU usage that are in evidence, the worse the overall performance it is. This type of data is usually seen in Performance Monitor collection sets. You must use the SQL query data to understand what applications, client computers, or SOLIDWORKS PDM operations are involved.

When analyzing a SQL query, it is important to identify the name of the SOLIDWORKS PDM stored procedures. In most cases, these stored procedures have a self-explanatory name. For example, ***XRef_GetTree*** means getting file references, ***SPA_AddDocuments*** means the operation to add documents, and so forth.

If a SOLIDWORKS PDM vault stored procedure appears in a SQL event that takes a long duration, or is involved in a deadlock or blocking process report, the stored procedure helps identify the cause of the performance issue.

SQL queries that have stored procedures may include as parameters document IDs or user IDs, which you might use to reproduce the issue. Eventually, this should help solve the problem.



```
go
If (object_id N'tempdb..#CounterIDs') Is Not Nul   ncate Table #Coun
If (object_id(N'tempdb..#XRefRoots') Is Not Null) Truncate Table #XRefRoo
Int,Config nvarchar (255) collate database_default ,ItemName nvarchar (255
SET NOCOUNT ON

Insert Into #XRefRoots
Select 4, 0, N'1899-12-30T00:00:00.000', 1, N'', N'', 0
Exec dbo.XRef_GetTree 0, 2, 1, N'f70051ae-a298-4805-bb42-187669749c9c', 1
go
Select ReactorTableVersion From SystemInfo

go
Exec Arc_GetFileStatus 2,4,1,3, 11
go
```

Ready.

The next section covers additional information about how to analyze performance data.

c) Performance Data Collection Setup

i) SQL Server tracing

You can use SQL Server tracing to get detailed information about database server performance. For SOLIDWORKS PDM, you can use tracings to find information such as slow running operations (based on queries) or random performance incidents. There are two main types of SQL tracing that SOLIDWORKS Technical Support recommends:

- To capture SQL workload – This method is useful when reproducing an issue, or if you need to understand the CPU breakdown by the main SOLIDWORKS PDM operations, applications, or client computers.
- To collect reports for random key performance indicators such as deadlocks or blocking processes.

For tracing, the recommendation is to use SQL Server Extended Events sessions that are available in SQL Server Management Studio and require SQL Server 2012 or newer.

For the SQL Server 2008 R2 versions, it is still possible to collect trace data by using tools such as *SQL Server Profiler*.

SQL Server Profiler is especially useful when reporting simple SOLIDWORKS PDM issues. For example, when viewing a Bill of Materials, or if a specific search is not performed in a timely manner.

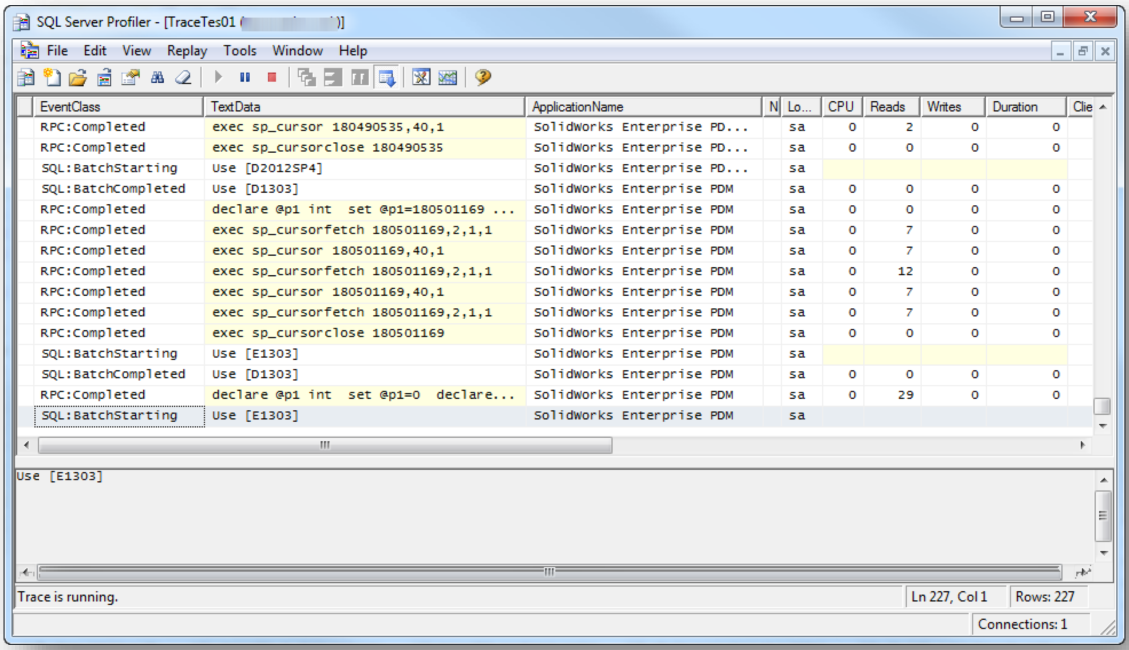
- **SQL workload tracing**

SQL Server workload helps provide an understanding of the flow of queries. As an example, it can identify transactions that take a long time, consume too much CPU, or that have an unusual number of reads. Depending on SQL Server versions, you should follow these steps to collect the SQL workload:

- **SQL Server 2008 R2**

For this SQL Server version and for any simple case that is easy to reproduce (for example, a specific file operation is very slow), use SQL Server Profiler and follow these steps:

1. Open SQL Server Profiler. Go to **Start** > **All Programs** > **Microsoft SQL Server** > **Performance Tools**. You can also access the tool from SQL Server Management Studio under **Tools** > **SQL Server Profiler**.
2. Click **File** > **New Trace** and connect to the SQL Server.
3. In the **Trace Properties** dialog box, select the **Standard** template in the **Use the template** list.
4. In the **Trace name** field, type a name for the trace. For example, ***TraceTest01***.
5. Select the **Save to File** option and specify a location in which to save the trace. Then click **Save**.
6. Change the **Set maximum file size** option to **50**. The default value of **5** MB is usually too small and may end up creating multiple trace files.
7. On the **Events Selection** tab, select the **Show all columns** option.
8. Right-click the **DatabaseName** header > **Select Column**. This column is useful later to filter the results based on databases.
9. On the **Events Selection** tab, right-click the **HostName** header > **Select Column**. This column is useful later to filter the results based on clients.
10. Click **Run**.


SQL Server Profiler - [TraceTes01 ([redacted])]

	EventClass	TextData	ApplicationName	N	Lo...	CPU	Reads	Writes	Duration	Clie
	RPC:Completed	exec sp_cursor 180490535,40,1	SolidWorks Enterprise PD...		sa	0	2	0	0	
	RPC:Completed	exec sp_cursorclose 180490535	SolidWorks Enterprise PD...		sa	0	0	0	0	
	SQL:BatchStarting	Use [D2012SP4]	SolidWorks Enterprise PD...		sa					
	SQL:BatchCompleted	Use [D1303]	SolidWorks Enterprise PDM		sa	0	0	0	0	
	RPC:Completed	declare @p1 int set @p1=180501169 ...	SolidWorks Enterprise PDM		sa	0	0	0	0	
	RPC:Completed	exec sp_cursorfetch 180501169,2,1,1	SolidWorks Enterprise PDM		sa	0	7	0	0	
	RPC:Completed	exec sp_cursor 180501169,40,1	SolidWorks Enterprise PDM		sa	0	7	0	0	
	RPC:Completed	exec sp_cursorfetch 180501169,2,1,1	SolidWorks Enterprise PDM		sa	0	12	0	0	
	RPC:Completed	exec sp_cursor 180501169,40,1	SolidWorks Enterprise PDM		sa	0	7	0	0	
	RPC:Completed	exec sp_cursorfetch 180501169,2,1,1	SolidWorks Enterprise PDM		sa	0	7	0	0	
	RPC:Completed	exec sp_cursorclose 180501169	SolidWorks Enterprise PDM		sa	0	0	0	0	
	SQL:BatchStarting	Use [E1303]	SolidWorks Enterprise PDM		sa					
	SQL:BatchCompleted	Use [D1303]	SolidWorks Enterprise PDM		sa	0	0	0	0	
	RPC:Completed	declare @p1 int set @p1=0 declare...	SolidWorks Enterprise PDM		sa	0	29	0	0	
	SQL:BatchStarting	Use [E1303]	SolidWorks Enterprise PDM		sa					

```
Use [E1303]
```

Trace is running. | Ln 227, Col 1 | Rows: 227

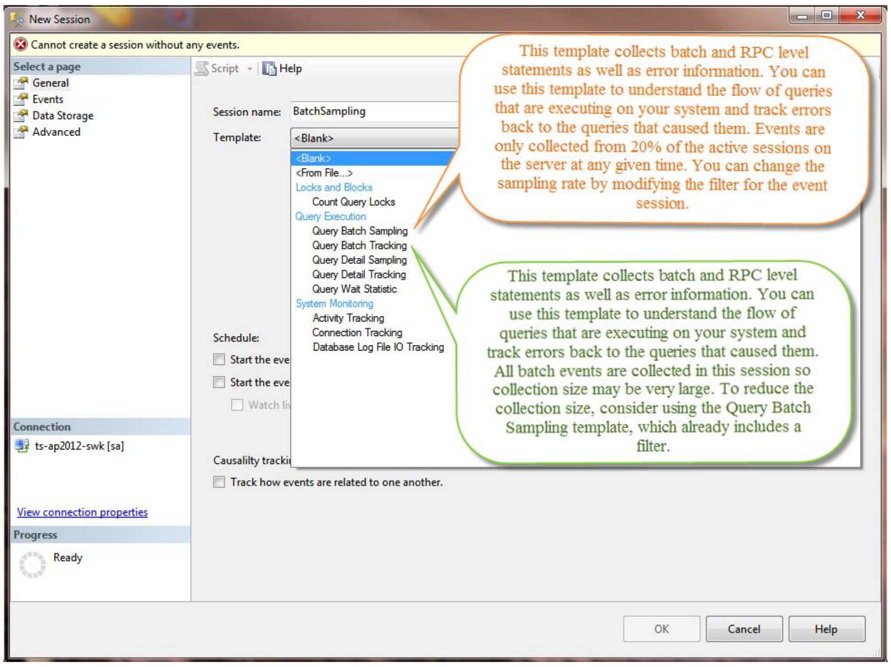
Connections: 1

11. If the task is to obtain a SQL workload sample, allow the trace to run for 20 minutes. However, if the reason to run the trace is for a specific issue, perform the action that causes performance problem. For example, check in the file to which you previously browsed.

12. After completing the action that has a performance problem, click **Stop Selected Trace** (the red stop button) in the SQL Server Profiler.

- **For SQL Server 2012 or newer**

Use Extended Events sessions based on the **Query Batch Tracking** template that tracks all active sessions.



New Session

Cannot create a session without any events.

Session name: BatchSampling

Template: <Blank>

<Blank>
<From File...>
Locks and Blocks
Count Query Locks
Query Execution
Query Batch Sampling
Query Batch Tracking
Query Detail Sampling
Query Detail Tracking
Query Wait Statistic
System Monitoring
Activity Tracking
Connection Tracking
Database Log File IO Tracking

Track how events are related to one another.

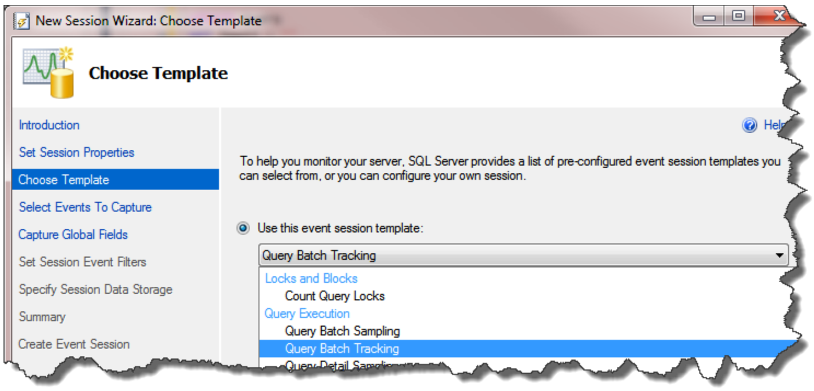
Connection: ts-ap2012-swk [sa]

This template collects batch and RPC level statements as well as error information. You can use this template to understand the flow of queries that are executing on your system and track errors back to the queries that caused them. Events are only collected from 20% of the active sessions on the server at any given time. You can change the sampling rate by modifying the filter for the event session.

This template collects batch and RPC level statements as well as error information. You can use this template to understand the flow of queries that are executing on your system and track errors back to the queries that caused them. All batch events are collected in this session so collection size may be very large. To reduce the collection size, consider using the Query Batch Sampling template, which already includes a filter.

To configure a SQL Server workload session, follow these steps:

1. Open SQL Server Management Studio.
2. Connect to the production server instance.
3. Expand the SQL Server instance > **Management** > **Extended Events**.
4. Right-click **Sessions** > **New Session Wizard**.
5. Type a new name for the session. For example, **Query Batch Tracking**.
6. Click **Next** and then select the **Use this event session template** option.
7. Select the **Query Batch Tracking** template, and then click **Next**.



New Session Wizard: Choose Template

Choose Template

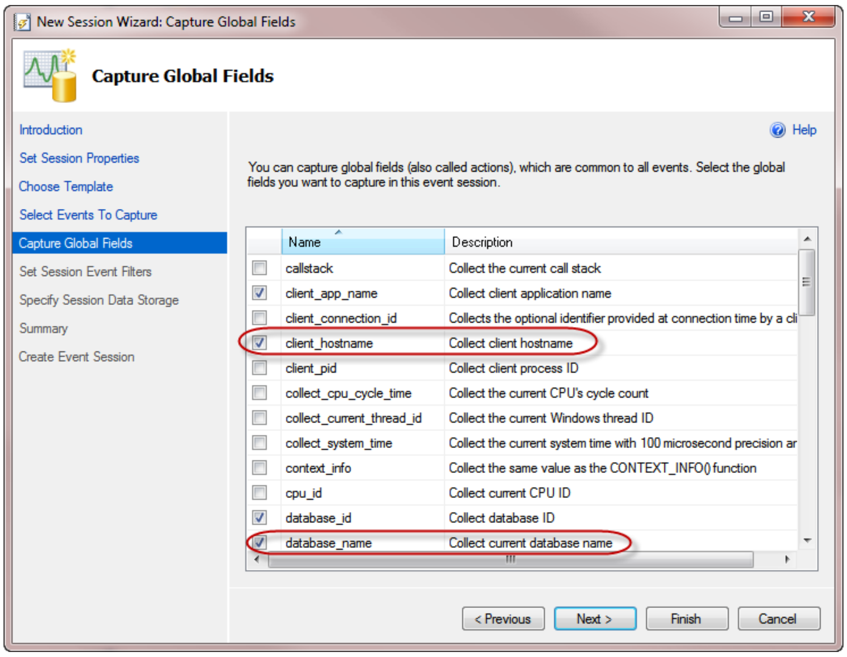
To help you monitor your server, SQL Server provides a list of pre-configured event session templates you can select from, or you can configure your own session.

Use this event session template:

Query Batch Tracking

Locks and Blocks
Count Query Locks
Query Execution
Query Batch Sampling
Query Batch Tracking

8. On the **Capture Global Fields** page, select the check boxes for the **client_hostname** and **database_name** fields.


New Session Wizard: Capture Global Fields

Capture Global Fields

- Introduction
- Set Session Properties
- Choose Template
- Select Events To Capture
- Capture Global Fields
- Set Session Event Filters
- Specify Session Data Storage
- Summary
- Create Event Session

Help

You can capture global fields (also called actions), which are common to all events. Select the global fields you want to capture in this event session.

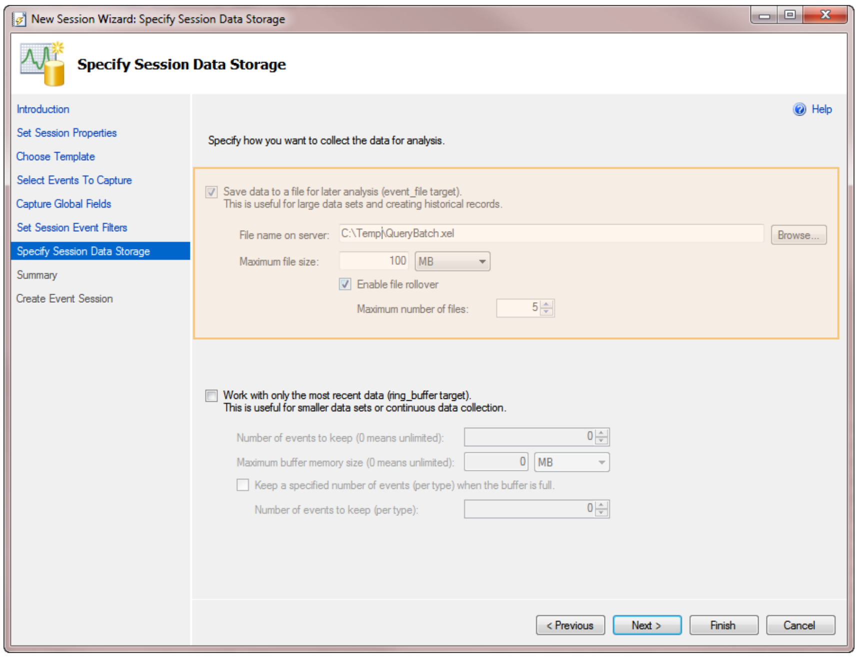
	Name	Description
☐	callstack	Collect the current call stack
✓	client_app_name	Collect client application name
☐	client_connection_id	Collects the optional identifier provided at connection time by a cli
✓	client_hostname	Collect client hostname
☐	client_pid	Collect client process ID
☐	collect_cpu_cycle_time	Collect the current CPU's cycle count
☐	collect_current_thread_id	Collect the current Windows thread ID
☐	collect_system_time	Collect the current system time with 100 microsecond precision ar
☐	context_info	Collect the same value as the CONTEXT_INFO() function
☐	cpu_id	Collect current CPU ID
✓	database_id	Collect database ID
✓	database_name	Collect current database name

9. Click **Next** and proceed to the **Specify Session Data Storage** page. Then select the **Save data to a file for later analysis** option.

10. For the **File name on server** option, browse to a file location in which you wish to save the trace file. For example, `C:\Temp\ QueryBatchTracking.xel`.

If you do not specify a file location, the trace file saves in the default location of the **Log** folder. For example, `C:\Program Files \Microsoft SQL Server \MSSQL[version].MSSQLSERVER\MSSQL\Log\ QueryBatchTracking.xel`. However, sometimes there is a change to the default location. Therefore, it is better to specify the location in which to save the trace file.

11. Ensure that the **Enable file rollover** option is active and that the total stored data is approximately 500 MB. For example, **Maximum file size** = **100 MB** and **Maximum number of files** is **5**. A size of 500 MB is a recommendation only. You can add more storage as needed.



Specify how you want to collect the data for analysis.

[✓] Save data to a file for later analysis (event_file target).
This is useful for large data sets and creating historical records.

File name on server: C:\Temp\QueryBatch.xel

Maximum file size: 100 MB

[✓] Enable file rollover

Maximum number of files: 5

[ ] Work with only the most recent data (ring_buffer target).
This is useful for smaller data sets or continuous data collection.

Number of events to keep (0 means unlimited): 0

Maximum buffer memory size (0 means unlimited): 0 MB

[ ] Keep a specified number of events (per type) when the buffer is full.

Number of events to keep (per type): 0

12. Click **Finish** and then click **Close**.

13. In the SQL Server Management Studio **Object Explorer**, expand the SQL Server instance and navigate to **Management > Extended Events > Sessions**.

14. Right-click **Query Batch** (or the name you specified in step 5) > **Start Session**.

15. After reproducing the issue or after 20 minutes when taking SQL workload samples, stop this extended session. To do this, repeat step 13, and then right-click **QueryBatchTracking** (or the name you specified in step 5) > **Stop Session**.

Important:

- All of these settings take effect immediately without stopping and restarting the server.

- After defining an Extended Events session, it is not mandatory to delete it. You can stop the session and keep it for use in the future.

- When troubleshooting a specific issue, you can use the **Query Batch Tracking** session example to start and stop while reproducing that specific issue. For performance cases where SQL Server CPU utilization is very high (for example > 80%-85%), the interval between starting and stopping a session should be a maximum of 5 minutes. However, if CPU usage exceeds 90%, you should not use this session.

- When sending trace files to Technical Support for analysis, the recommendation is to temporarily stop and restart the extended session. This ensures that the latest data capture writes to the Extended Events trace file. To do this, expand the SQL Server instance and navigate to **Management > Extended Events** > **Sessions** >

right-click **Query Batch** (or the name you specified in step 5) > **Stop Session**. Note that if want to continue using the session after you copy the trace file, make sure right-click the session > **Start Session**.

- You can also use the **Query Batch Sampling** template. This template captures by default only 20% of the active sessions, which means that you can run the session for intervals longer than just a few hours. However, data collected by this template is incomplete and not accurate enough. This template is mainly useful to get a general overview of the SQL workload and not to properly troubleshoot a performance issue. Therefore, avoid using this template unless SOLIDWORKS Technical Support provides explicit instruction to use it.

• Tracing deadlocks reports and blocked processes reports

For SQL Server 2008 R2, it is good to configure deadlocks reports and blocked processes reports as described in the Knowledge Base solution *S-069784*. This solution also includes additional SQL script files. Be aware that the output is trace files with the TRC file format.

For SQL Server 2012 or newer, the recommendation is to use server side Extended Events sessions because they are lighter and easy to configure to run on the server side. The following Extended Events sessions are recommended:

♦ Capturing locks and deadlocks using “system_health”

SQL Server 2012 and newer versions enable a **system_health** Extended Events session by default on. Among other things, this session captures deadlock reports and relevant database locks. To ensure that a system_health session captures performance data for several weeks or months, it is a recommendation to increase data storage for the session. To do this, follow these steps:

1. Open SQL Server Management Studio.

2. Connect to the production SQL Server instance.

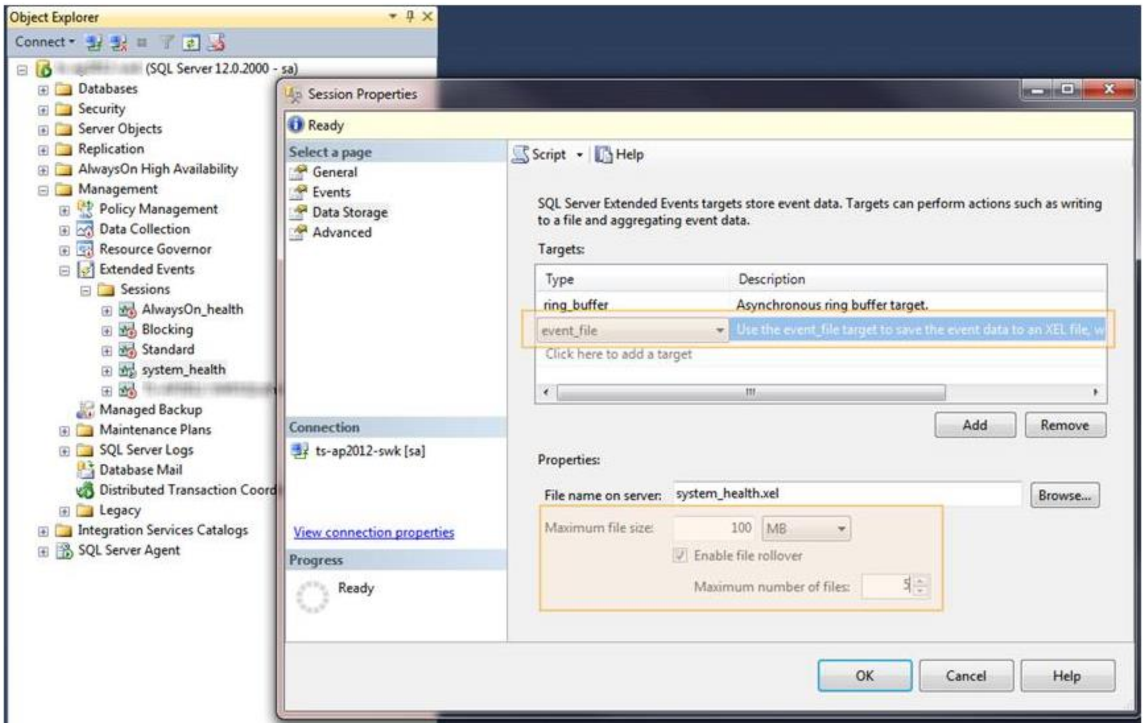
3. Expand the SQL Server instance and navigate to **Management** > **Extended Events** > **Sessions** > right-click **system_health** > **Properties**.

4. In the **Data Storage** section, click the **event_file** row and determine if **Maximum file size** is much greater than 5 MB. It is likely to specify use of the default setting that stores only 20MB of data.

5. If the total stored data is less than 500 MB, choose to **Remove** the **event_file.** The click **Add** and reselect **event_file** in the drop-down list.

6. Make sure that the **Enable file rollover** option is active and that the total stored data is approximately 500 MB. For example, **Maximum file size** = **100 MB** and

Maximum number of files is **5**. A size of 500 MB is a recommendation only. You can add more storage as needed.



SQL Server Extended Events targets store event data. Targets can perform actions such as writing to a file and aggregating event data.

Targets:

Type	Description
ring_buffer	Asynchronous ring buffer target.
event_file	Use the event_file target to save the event data to an XEL file, w
Click here to add a target	

Properties:

File name on server: system_health.xel

Maximum file size: 100 MB

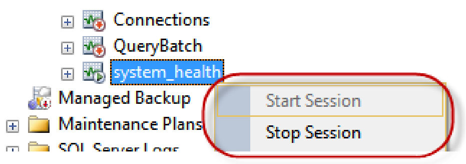
Enable file rollover: ✓

Maximum number of files: 5

7. Make sure that the **system_health** session is started.

Notes:

- When sending trace files to Technical Support for analysis, the recommendation is to temporarily stop and restart the extended session. This ensures that the latest data capture writes to the Extended Events trace file. To do this, expand the SQL Server instance and navigate to **Management** > **Extended Events** > **Sessions** > right-click **Query Batch** (or the name you specified in step 5) > **Stop Session**, and then right-click **Start Session**.

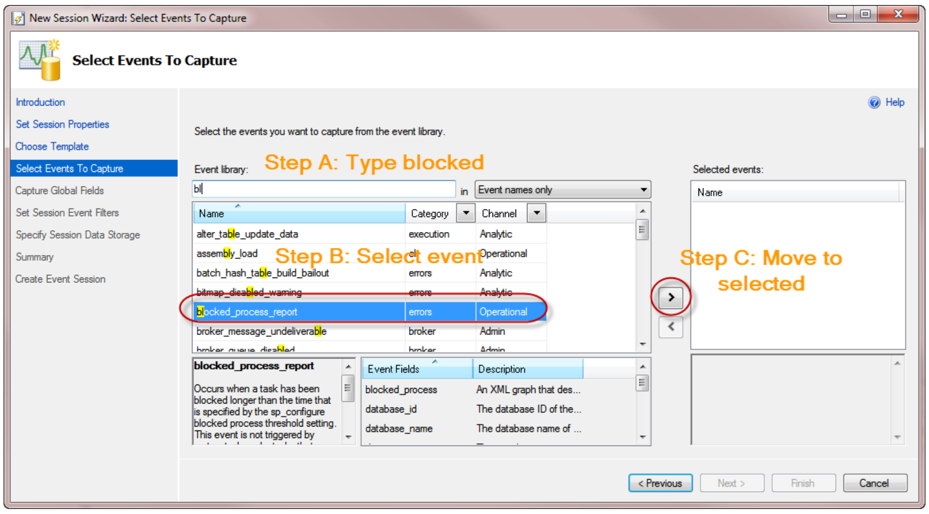


- The **system_health** extended session should remain on continuously because this is a default configuration on SQL Server.

♦ Capturing blocking locks

To capture blocking reports, follow these steps to create a new Extended Events session:

1. Open SQL Server Management Studio.
2. Connect to the production Server instance.
3. Expand the SQL Server instance and navigate to **Management > Extended Events**.
4. Right-click **Sessions > New Session Wizard**.
5. Type a name for the new session. For example, type **BlockingLocks**.
6. Click **Next** > select the **Do not use a template** option > **Next**.
7. On the **Select Events to Capture** page, double-click the **blocked_process_report** event (or press the arrow) to move the event to the **Selected Events** list.



Select the events you want to capture from the event library.

bl

Name	Category	Channel
alter_table_update_data	execution	Analytic
assembly_load	clr	Operational
batch_hash_table_build_bailout	errors	Analytic
bitmap_disabled_warning	errors	Analytic
blocked_process_report	errors	Operational
broker_message_undeliverable	broker	Admin
broker_queue_disabled	broker	Admin

blocked_process_report

Occurs when a task has been blocked longer than the time that is specified by the sp_configure blocked process threshold setting. This event is not triggered by

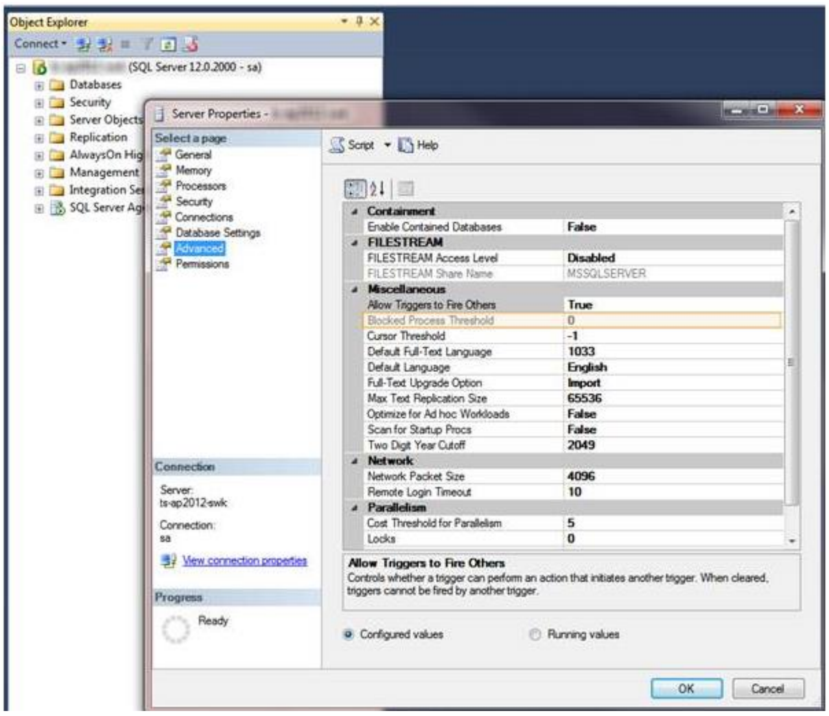
Event Fields	Description
blocked_process	An XML graph that des...
database_id	The database ID of the...
database_name	The database name of ...

8. Click **Next** and proceed to the **Specify Session Data Storage** page. Then select the **Save data to a file for later analysis** option.
9. For the **File name on server** option, browse to a file location in which you wish to save the trace file. For example, `C:\Temp\ BlockingLocks.xel`.

Note: If you do not specify a file location, the trace file will be saved in the default location of the 'Log' folder. For example, `C:\Program Files \Microsoft SQL Server \MSSQL[version].MSSQLSERVER\MSSQL\Log\ BlockingLocks.xel`. However, sometimes there is a change to the default location. Therefore, it is better to specify the location in which to save the trace file.

10. Click **Finish** and then click **Close**.

11. In **Object Explorer**, right-click the SQL Server instance > **Properties** > **Advanced** and set the **Blocked Process Threshold** event to 20 seconds. In test environments, you could use a value of 5 seconds instead.



Containment	
Enable Contained Databases	False
FILESTREAM	
FILESTREAM Access Level	Disabled
FILESTREAM Share Name	MSSQLSERVER
Miscellaneous	
Allow Triggers to Fire Others	True
Blocked Process Threshold	0
Cursor Threshold	-1
Default Full-Text Language	1033
Default Language	English
Full-Text Upgrade Option	Import
Max Text Replication Size	65536
Optimize for Ad hoc Workloads	False
Scan for Startup Procs	False
Two Digit Year Cutoff	2049
Network	
Network Packet Size	4096
Remote Login Timeout	10
Parallelism	
Cost Threshold for Parallelism	5
Locks	0

Allow Triggers to Fire Others
Controls whether a trigger can perform an action that initiates another trigger. When cleared, triggers cannot be fired by another trigger.

[✓ Configured values] Running values

12. Expand the SQL Server instance and navigate to **Management** > **Extended Events** > **Sessions** > **Blockings** (or the name specified in step 5) > **Start Session**.

Important:

- It is a best practice to use a blocking extended session only when you require server performance analysis. If after capturing several weeks of data there are no reasons to continue capturing blocking incidents, you should stop the session and set the blocked process threshold value to **0** (zero) to disable the session. You can do this from the **Advanced** tab of the SQL Server instance **Properties**.
- Under no circumstances should the values for the blocked process threshold be lower than **5**. If the value is lower than **20**, the process captures more blocking events. However, for very busy SQL Servers, there may be an indication to increase the threshold to 30 seconds.
- All of these settings take effect immediately without stopping and restarting the server.
- When sending trace files to Technical Support for analysis, the recommendation is to temporarily stop and restart the extended session. This ensures that the latest data capture writes to the Extended Events trace file. To do this, expand the SQL Server instance and navigate to **Management** > '**Extended Events** > **Sessions** > right-click **Query Batch** (or the name you specified in step 5) > **Stop Session**. Then, right-click **Start Session**. If you want to continue using the session after you copy the trace file, make sure to right-click the session > **Start Session**.

- **Analyzing SQL trace data**

You can use the SQL Server Profiler tool to review SQL traces with the TRC file format.

You can open Extended Events sessions in SQL Server Management Studio for analysis.

VARs can use for Extended Events sessions the *SOLIDWORKS Log Parser tool* (Reseller Version), which is available in the Knowledge Base solution *S-067430*. One of the advantages of using the Log Parser tool is that it can generate charts and extract a blocking chain details report. This section covers only the following main Extended Events that the Log Parser tool supports:

- For SQL workload analysis: ***rpc_completed*** and ***sql_batch_completed***.
- For key performance indicators such as database locks, deadlocks or blocking chains: ***wait_info***, ***xml_deadlock_report*** and ***blocked_process_report***.

- **Windows Performance Monitor tool**

Windows provides the Performance Monitor tool that you can use to record specific aspects (counters) of the system performance over long periods or time. You can start this tool from the command line by entering **perfmon**. The SQL Server installation adds other counters that are accessible with this tool.

If SQL Server counters are not available when setting up this collection set, this might be an indication that the counters are inactive or that there was a conflict during SQL Server installation or update. In most cases, running a SQL Server Repair operation restores the missing SQL counters.

- **Monitoring performance at all times**

When capturing data over long time intervals (such as several weeks), the recommendation is to avoid using a Performance Monitor data set with too many counters. For example, the counters in the following list should be enough to capture the main key performance indicators without creating significant overhead on SQL Server:

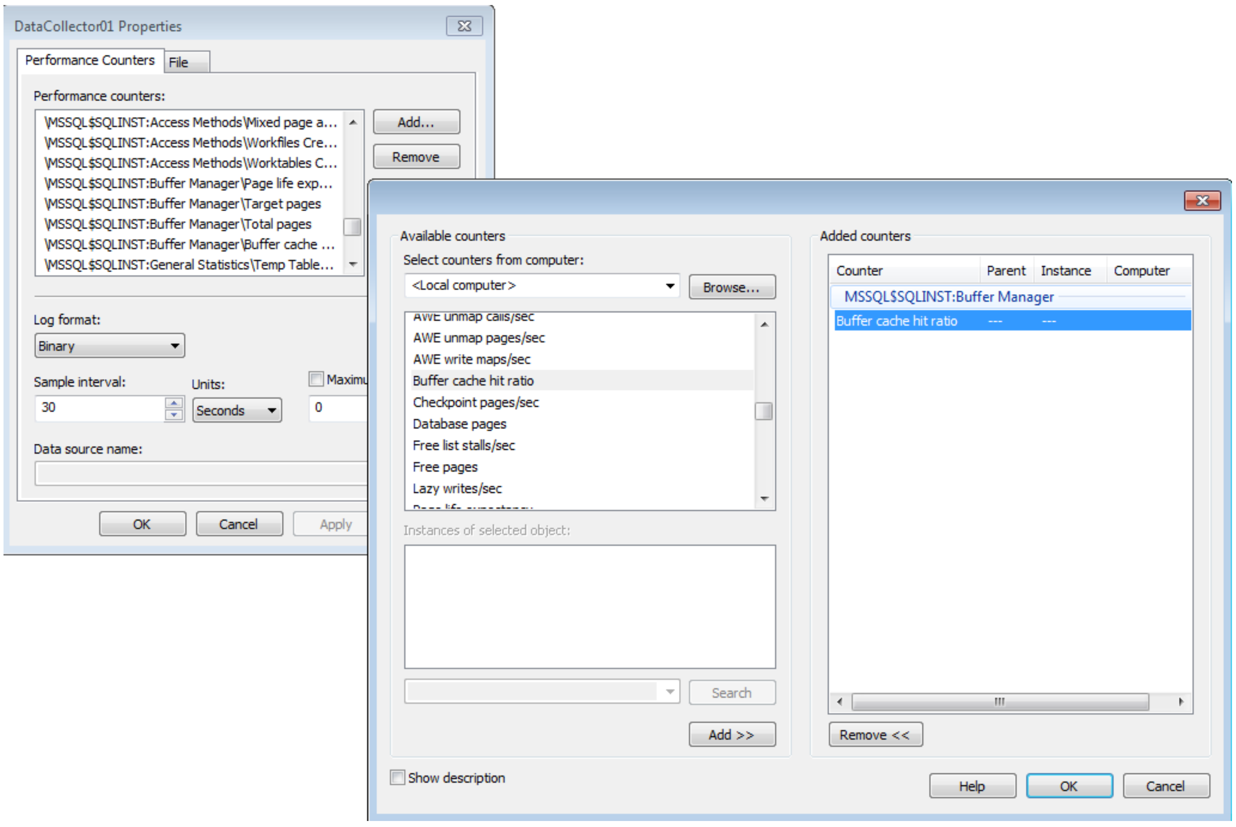
- Memory: Available Mbytes
- Processor: Processor Time\ _Total
- SQLServer: General Statistics\Processes Blocked
- SQLServer: General Statistics\User Connections
- SQLServer: Locks\Number of Deadlocks/sec

The **Data Collector Properties** include a specification for the sample interval. Consider defining a time interval sampling that is equal to or greater than 1 minute.

- ## Custom Performance Monitor collection data sets

Many more counters are available in Performance Monitor. A guide about which counters are most useful when investigating SQL Server performance issues is available in the Knowledge Base solution *S-025698*. This solution includes many counters with a short time interval sampling that increases the resource usage on SQL Server. These counters in this solution are not appropriate for use over long time intervals.

For more complex performance issues, you can select the appropriate counters in this solution to define a Data Collector for Performance Monitor.


DataCollector01 Properties

Performance Counters | File

Performance counters:

- \MSSQL$SQLINST:Access Methods\Mixed page a...
- \MSSQL$SQLINST:Access Methods\Workfiles Cre...
- \MSSQL$SQLINST:Access Methods\Worktables C...
- \MSSQL$SQLINST:Buffer Manager\Page life exp...
- \MSSQL$SQLINST:Buffer Manager\Target pages
- \MSSQL$SQLINST:Buffer Manager\Total pages
- \MSSQL$SQLINST:Buffer Manager\Buffer cache ...
- \MSSQL$SQLINST:General Statistics\Temp Table...

Log format: Binary

Sample interval: 30 Units: Seconds

Maximu 0

Data source name:

Available counters

Select counters from computer: <Local computer>

- AWE unmap calls/sec
- AWE unmap pages/sec
- AWE write maps/sec
- Buffer cache hit ratio
- Checkpoint pages/sec
- Database pages
- Free list stalls/sec
- Free pages
- Lazy writes/sec
- Page life expectancy

Instances of selected object:

Added counters

Counter	Parent	Instance	Computer
MSSQL$SQLINST:Buffer Manager			
Buffer cache hit ratio	---	---	

Show description

The following sections provide guidelines for what some might consider “normal” or “”ideal” values for these counters, and what might indicate a problem area. Do not consider these values as absolute. You can obtain the most useful information by observing how these counters changeover time.

♦ **Processor counters**

- **Processor=>*%Processor Time: _Total*** – This is the percentage of elapsed time the processor spends running non-idle threads. Ideally < 80%. This is only useful on physical machines. The value drops dramatically on virtual machines because you cannot see the cause of CPU loading.

- **Processor=>*%Privileged Time: _Total*** - This is the amount of time the processor was busy with Kernel mode operations. If the processor is very busy and this mode is high, it is usually an indication of some type of OS service having difficulty, although user mode programs can make calls to the Kernel mode OS components to cause this type of performance issue occasionally.
- **Processor=>*User Time: _Total*** – The value of this counter helps to determine the kind of processing that is affecting the system. This is the total amount of non-idle time that was spent on user mode operations, generally this means application code.
- **Processor=>*Interrupts/sec*** – This is the number of interrupts the processor was asked to respond to. Interrupts are generated from hardware components such as hard disk controller adapters and network interface cards. A sustained value over 1000 usually indicates a problem. Problems include poor configuration of drivers, errors in drivers, excessive utilization of a device (such as a network interface controller (NIC) on an IIS server), or hardware failure. Compare this value with the ***System=>Systems Calls/sec*** counter. If the ***Interrupts/sec*** is much larger over a sustained period, there is probably a hardware issue.
- **Process(sqlservr)=>*% Process Time*** – (on the SQLSERVR instance) This is the percentage of processor time spent on SQL Server process threads. Ideally <80%.
- **Process(sqlservr)=>*Private Bytes***
- **Process(sqlservr)=>*Working Set***
- **Server=>*Pool Nonpaged Failures***
- **Server=>*Pool Paged Failures***
- **Server=>*Pool Nonpaged Peak***
- **System=>*Context Switches/sec***
- **System=>*Processor Queue Length*** – This is the number of threads waiting for CPU cycles, where < 12 per CPU is good to fair. Fewer than 8 is better. Fewer than 4 is best.

♦ Memory and Pagefile usage counters

Available memory is usually the most common source for performance issues on a Windows Server installation. Fortunately, this is an easy metric to measure because there are several counters in the memory object that can help troubleshoot memory issues. Most notably, two important counters provide a reasonably accurate overview of memory pressures. These are the ***Page Faults/sec*** and ***Pages/sec*** counters. Using these two memory counters alone can highlight if the system is correctly configured and experiencing memory issues. The following counters are recommended to monitor memory and ***pagefile.sys*** file (pagefile) usage.

- **Memory=>*Available Mbytes*** – Shows the unused physical memory (not using page file).
- **Memory=>*Cache Bytes***

- **Memory=>*Commit Limit***
- **Memory=>*CommittedBytes*** – This counter monitors the amount of memory (in bytes) that has been allocated by the various processes. As this increases above the available memory, so does the pagefile size because the paging has increased.
- **Memory=>*Page Input/sec*** – Shows the virtual memory pages read from the hard disk per second. This can help to identify paging as a bottleneck. The value should be less than 10.
- **Memory=>*Pages/sec*** – Shows the number of pages that are read from or written to the disk.
- **Memory=>*Page Faults/sec*** – Gives a high-level view of memory related performance problems.
- **Memory=>*Pool Nonpaged Bytes***
- **Paging File=>*%Usage*** - Shows the percentage of the paging file that is actually in use. You can use this counter to determine if the Windows pagefile is a potential bottleneck. If this value consistently remains above 50% or 75%, you should consider increasing the pagefile size or moving the pagefile to another disk.
- **Paging File=>*%Usage Peak*** – This is the highest %Usage metric since the last server restart.

The most important memory counter is ***Pages/sec***, which shows the number of pages read from or written to disk. Therefore, this counter is a representation of the quantity of hard page faults that the system is experiencing. Microsoft recommends upgrading the amount of memory if the ***Pages/sec*** value consistently averages more than 5 pages per second. In practice, you will probably begin to experience slower system performance only when the value is consistently over 20. Therefore, it is good to watch this counter carefully once it pushes over 10 pages per second.

The ***Pages/sec*** counter is also useful in determining if the system is “thrashing,” that is, experiencing more than 100 pages per second. You should prevent thrashing from occurring on a Windows Server 2008 R2 system because the reliance on the disk to resolve memory faults dramatically affects how efficiently the system can cope with workloads.

A system experiences page faults when a process requires data or code that it cannot find in its *working set*. A working set is the amount of memory committed to a certain process. When a process must retrieve the data or code in another part of physical memory, this is a “soft fault”. A “hard fault” refers to a process that retrieves the data or code from the disk subsystem. Modern systems can cope with a large number of soft faults without significant performance impairment. However, because a hard fault requires disk subsystem access, this causes the process to wait a significant amount. This dramatically slows performance. The **Memory** section of the **Resource Monitor** in Performance Monitor includes columns that display working sets and hard faults.

The ***Page Faults/sec*** counter displays both hard and soft faults. This counter often displays rather high numbers that sometimes exceed several hundred faults per second. When the counter exceeds several hundred page faults per second for extended durations, you should check other memory counters to try to identify if a bottleneck exists. The system memory is limited in size, and Windows Server supplements the installed RAM with virtual memory, which is more abundant. Windows begins paging to disk when the entire installed RAM is in use. This in turn frees the RAM for new processes and applications.

Windows generally increases the size of the pagefile automatically as necessary. In certain cases however, you may want to tune the system performance and manage the virtual memory settings yourself. You can significantly improve performance by maintaining the default pagefile in the system drive, and then adding another pagefile to an additional hard drive. Spanning virtual memory across several disks or simply placing the pagefile on a second, less frequently accessed disk can also boost the performance of your Windows Server installation. However, you need to ensure that the secondary disk is not slower than the disk that currently holds the pagefile.

♦ Physical Disk usage counters

The most useful counters for monitoring the disk subsystem are the ***% Disk Time*** and ***Avg. Disk Queue Length*** counters.

- **PhysicalDisk=>*% Disk Time*** – This counter monitors the time that a certain physical or logical drive uses in servicing the read and write requests.
- **PhysicalDisk=>*Average Disk Queue Length*** – This counts the number of unserviced requests on the physical or logical drive. The *Avg. Disk Queue Length* is an interval average and therefore is a numerical representation of the number of delays the disk drive experiences. In general, if the delay often exceeds **2**, the disks are inadequate to service the system workload and performance may be compromised.
- **PhysicalDisk=>*Average Disk sec/Read* –** This counter is a key measure of disk latency. It represents the average time, in milliseconds, of each read from disk**.** A value of **8 ms** is good.
- **PhysicalDisk=>*Average Disk sec/Write*** – This counter is a key measure of disk latency. It represents the average time, in milliseconds, of each write to disk**.** A value of > **20 ms** non-cached is poor.
- **PhysicalDisk=>*Disk Reads/sec***
- **PhysicalDisk=>*Disk Write Bytes/sec***
- **PhysicalDisk=>*Avg. Disk sec/Transfer*** – This counter relates to the ***pagefile.sys*** file.

♦ SQL Server counters

- **Access Methods=>*Mixed page allocations/sec***
- **Access Methods=>*Workfiles Created/sec*** – This is the number of work files created per second. Tempdb workfiles are used in processing hash operations when the amount of data being processed is too big to fit into the available memory. You might be able to reduce this number by making the queries more efficient by adding or changing indexes, adding additional memory, etc.
- **Access Methods=>*Worktables Created/sec*** – This is the number of worktables created in tempdb per second. Worktables are used for queries that use various spools (table spool, index spool, and others).
- **Buffer Manager=>*Page life expectancy*** – This is the number of seconds a page stays in the buffer pool without references. A buffer that has a 300-second page life expectancy keeps any given page in memory in the buffer pool for 5 minutes before the buffer pool flushes the page to disk, unless a process references the page. Microsoft recommends 300 seconds as the minimum target for page life expectancy.
- **Buffer Manager=>*Target pages***
- **Buffer Manager=>*Total pages***
- **Buffer Manager=>*Buffer cache hit ratio*** – This counter indicates how often SQL Server goes to the buffer, not the hard disk, to get data. The higher the ratio, the less often SQL Server must go to the hard disk to fetch data. Overall, this boosts performance. A ratio of 90% to 95% is ideal. In addition, watching how this ratio changes over time can give you a good idea of how the system performance is likely to affect users.

  Unlike many of the other counters available for monitoring SQL Server, this counter averages the buffer cache hit ratio from the time of the last SQL Server restart. In other words, this counter is not a real-time measurement, but an average of all of the days since the last SQL Server restart. Because of this, if you want to get an accurate record of what is happening in your buffer cache right now, you must stop and then restart the SQL Server service. Afterward, allow SQL Server to run for several hours of normal activity before checking this figure.

- **General Statistics=>*Temp Tables Creation Rate***
- **General Statistics=>*Temp Tables For Destruction***
- **General Statistics=>*User Connections*** – This counter does not appear in the Solution document. However, it can be a useful counter to evaluate how the number of users correlates with other performance factors.
- **SQL Statistics=>*Batch Requests/sec*** - This counter is a good general indicator for the activity level of the SQL Server. The counter is highly dependent on the hardware and the nature of the application running on the server.

- **SQL Statistics=>*SQL Compilations/sec*** – This is the number of times that Transact-SQL compilations occur, per second (including recompiles). A lower value is better and should be less than 10% of the ***Batch Requests/sec*** counter.
- **SQL Statistics=>*SQL Re-Compilations/sec*** – This is the number of times, per second, that Transact-SQL objects attempted to run but had to be recompiled before completion. This number should be at or near zero because recompiles can cause deadlocks and exclusive compile locks.
- **Transactions=>*Free Space in tempdb (KB)***
- **Locks=>*Number of Deadlocks/sec*** – This is the number of lock requests, per second, that resulted in a deadlock. Because only a COMMIT, ROLLBACK, or deadlock can terminate a transaction (excluding failures or errors), it is important to track this value.
- **Memory Manager=>*Memory Grants Pending* -** This counter shows the current number of processes that are waiting for a workspace memory grant. The target value should be close to **0**.

♦ How to interpret performance metrics for disks

For a logical drive on which the OS is installed, the instance of counters is not important. There is nothing you can do about how the OS accesses the drive, and apart from startup, it is not used heavily. If you have not yet moved some of the SQL Server database files (like tempdb) to a different logical drive, there might be a lot of activity for these counters.

The logical drives that you are most interested in are those that hold the data MDF and NDF files for SQL Server. The amount of drive activity relates directly to the amount of RAM that is available for SQL Server to use.

If the overall database size is less than the amount of RAM that is available to SQL Server, the database pages are read gradually from the disk into memory. There will then be very little disk activity. This is because the SELECT (read) queries will access the data from RAM.

If the overall database size is larger than the RAM that is available to SQL Server, the disk activity reflects the paging of data from the disk into RAM and RAM to the disk.

In the second case, the ***% Disk Time*** counter is a reflection of the ratio of the database size to RAM. A high ratio (large database and small amount of RAM) will cause a high ***% Disk Time*** if the disk subsystem cannot keep up with the read and write requests of the database. Whether the disk drive can keep up is relative to the amount of usage, the mix of queries to the database, and the configuration of the disk subsystem.

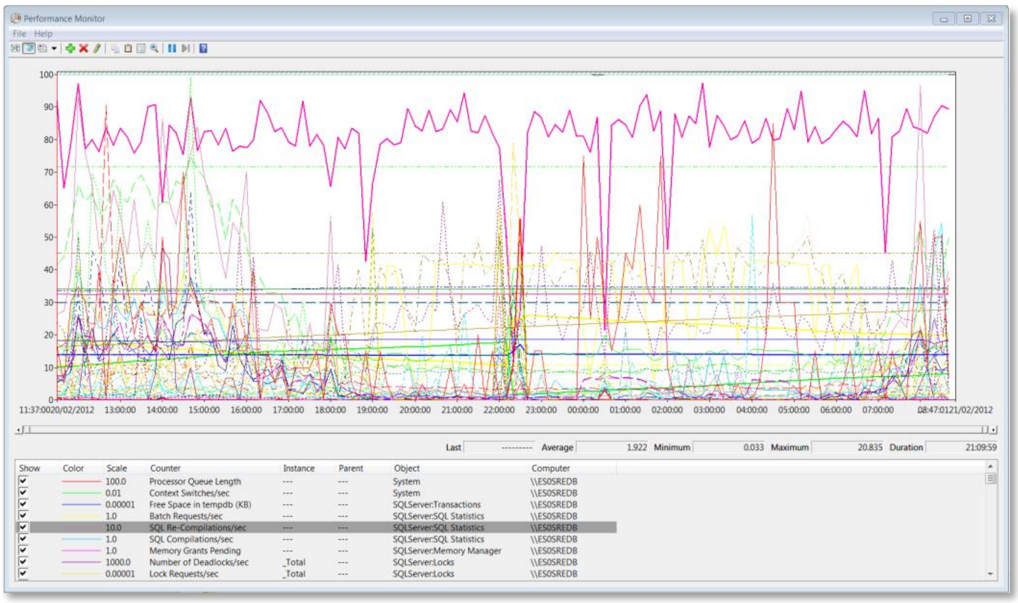
If ***% Disk Time*** averages over 70 percent, you need to check the second counter for that logical drive's ***Current Disk Queue Length***. This is an indication of the pending “waits” for the spindle. The term “waits” means that SQL Server was delayed accessing data from the disk. The ***Current Disk Queue Length*** value should not exceed twice the number of spindles in the logical drive. In a RAID 1 configuration with two drives (one spindle each), you want a maximum ***Current Disk Queue Length*** value of ***4***.

If you experience high ***Current Disk Queue Length***, consider the following:

- Use faster disk drives.
- Add another RAID 1 array and create data files on that array for each filegroup on the system.
- If using RAID 1+0 or RAID 5, you can add additional disks to increase the spindles.

- **Analysis of Performance Monitor logs**

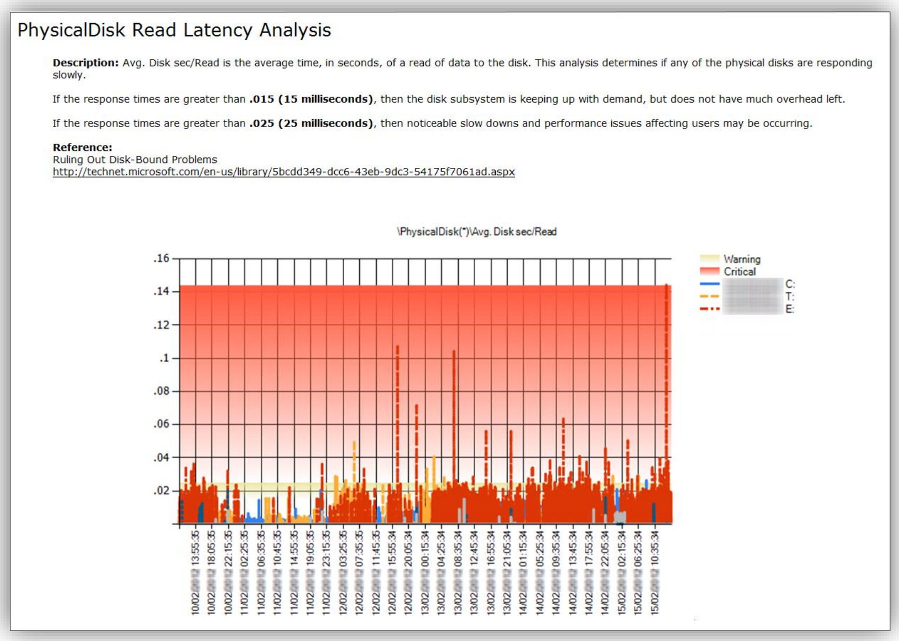
It is possible to use Performance Monitor to view and analyze the captured log files. Various views are possible, and you can switch different sets of counters on or off to focus on the different aspects of the analysis.



Last --------- Average 1.922 Minimum 0.033 Maximum 20.835 Duration 21:09:59

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	100.0	Processor Queue Length	---	---	System	\\ES0SREDB
✓	🟩	0.01	Context Switches/sec	---	---	System	\\ES0SREDB
✓	🟦	0.00001	Free Space in tempdb (KB)	---	---	SQLServer:Transactions	\\ES0SREDB
✓	🟨	1.0	Batch Requests/sec	---	---	SQLServer:SQL Statistics	\\ES0SREDB
✓		10.0	SQL Re-Compilations/sec	---	---	SQLServer:SQL Statistics	\\ES0SREDB
✓	🟦	1.0	SQL Compilations/sec	---	---	SQLServer:SQL Statistics	\\ES0SREDB
✓	🟪	1.0	Memory Grants Pending	---	---	SQLServer:Memory Manager	\\ES0SREDB
✓	🟪	1000.0	Number of Deadlocks/sec	_Total	---	SQLServer:Locks	\\ES0SREDB
✓	🟨	0.00001	Lock Requests/sec	_Total	---	SQLServer:Locks	\\ES0SREDB

It is also possible to acquire third-party tools to help you review the captured data. One convenient tool is the Performance Analysis of Logs (PAL) utility, which is an open source project that available under a Microsoft Public License. The PAL tool applies guidelines that are relevant to specific applications (such as SQL Server) to help highlight counters that exceed recommended values.


PhysicalDisk Read Latency Analysis

Description: Avg. Disk sec/Read is the average time, in seconds, of a read of data to the disk. This analysis determines if any of the physical disks are responding slowly.

If the response times are greater than **.015 (15 milliseconds)**, then the disk subsystem is keeping up with demand, but does not have much overhead left.

If the response times are greater than **.025 (25 milliseconds)**, then noticeable slow downs and performance issues affecting users may be occurring.

Reference:
Ruling Out Disk-Bound Problems
http://technet.microsoft.com/en-us/library/5bcdd349-dcc6-43eb-9dc3-54175f7061ad.aspx

You can find the PAL tool at *https://github.com/clinthuffman/PAL.*

d) Performance Baselines

One of the most difficult aspects of performance is trying to identify and quantify when the system is actually running "slowly", and whether the slowness relates to server activities.

To address this, SOLIDWORKS Technical Support recommends that you monitor and record a performance baseline of the SQL Server within two weeks after deployment. The length of capture can be two or three typical workdays. This establishes a benchmark against which you can compare any later investigations.

It is often worth waiting a few days for the system to "settle in". The latest versions of SQL Server have an element of automatic tuning that responds to typical usage patterns. In addition, you have an opportunity to resolve any teething problems.

You should then take equivalent measurements, perhaps every quarter. This allows you to make comparisons to the baseline measurements. It also means that you are prepared if you experience a sudden performance problem.

Measuring your system over time allows you to acquire the knowledge of what constitutes normal behavior, and what indicates an abnormal load, or degraded performance.

i) How to make performance baselines

The easiest way to generate performance baseline data is to run the SOLIDWORKS PDM Status Report tool once a day during working hours. To run the report, go to **File** > **Run Diagnostics and Generate Performance Snapshot Reports**.

For a more detailed baseline, collect the following data for two weeks:

- Windows Performance Monitor data as explained in the section "*Monitoring Performance At All Times*."
- Key performance indicators for random performance as explained in the section "*Tracing deadlocks reports and blocked processes reports*."
- Take two SQL Workload samples, each from a different weekday as explained in the section "*SQL Workload tracing*."

  Note that a SQL Workload sample data set is more of a requirement when you find a performance issue and want a more in depth assessment.

ii) Schedule monitoring

Repeat the performance capture every few months and compare it to the baseline. This becomes a proactive approach to maintaining peak performance.

iii) Compare performance benchmark data

Compare current performance metrics to the previous benchmark metrics and look for patterns that might indicate the source. This might lead you straight to a specific problem. However, it is often a combination of factors. Therefore, you need to investigate each of the different areas.

If you have not yet applied some recommendations from the "*Installation and Configuration*" chapter, when performance becomes an issue it is time to determine if it would be useful to apply any of those techniques.

e) Database Replication

For database replication, most of the troubleshooting guidelines still apply. However, it is a best practice to configure the recommended data collection sets for the primary site and for at least one secondary site.

f) Memory

Use Performance Monitor and **Status Report** > '**Performance** > **Memory Report** to look for signs of conflicts over RAM.

Consider:

- Moving applications off of the SQL Server hardware
- Disabling redundant OS services
- Disabling redundant SQL Server services
- Adding more RAM

g) Disk

Use Performance Monitor to look for disk hotspots.

Consider:

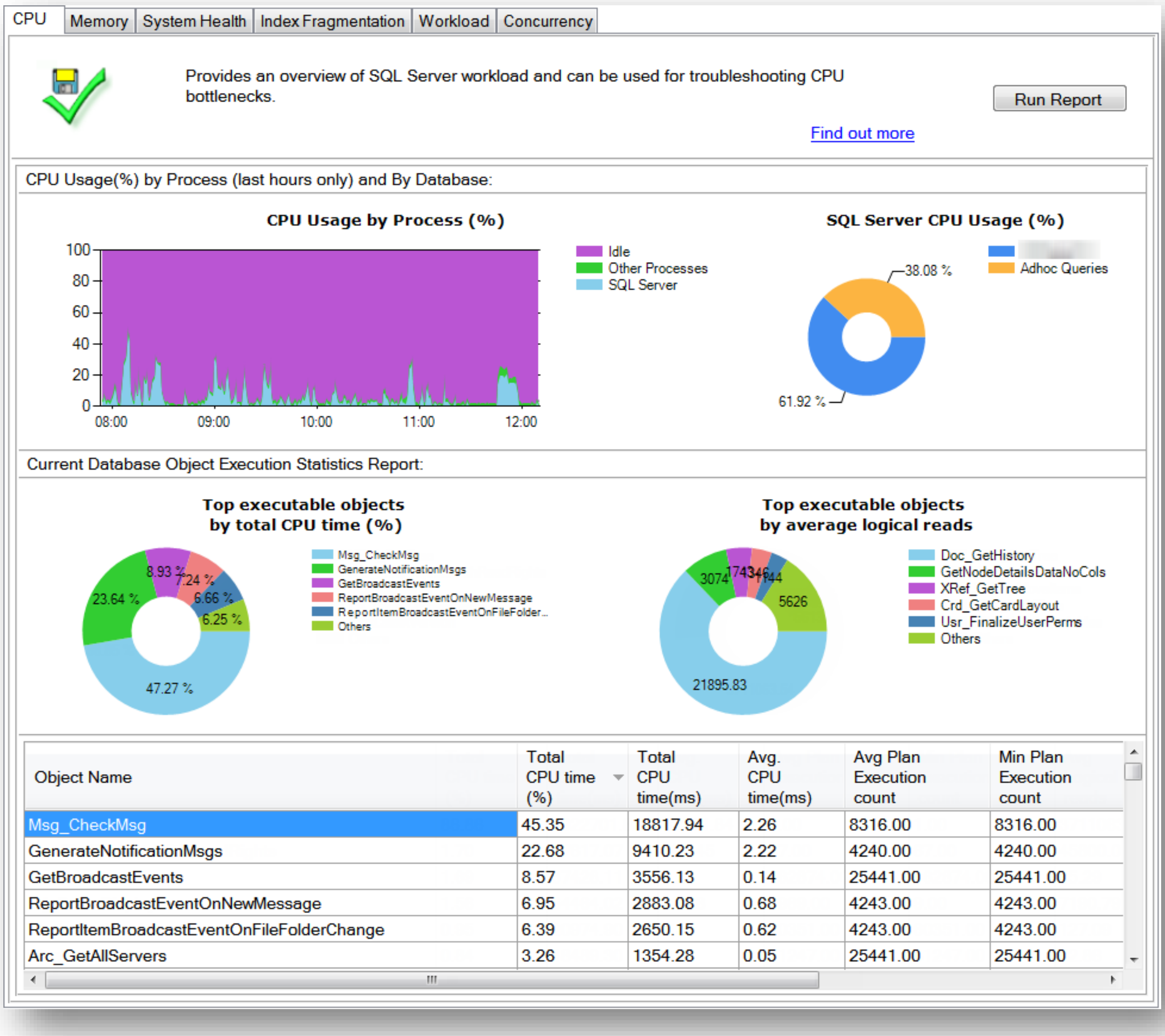
- Moving file locations
- Adding more data files
- Reconfiguring disks
- Adding more disks

h) Processor

Use Performance Monitor, Extended Events Sessions, and **Status Report** > **Performance** > **CPU** to see if the processors are the bottleneck.

Consider:

- Moving applications off of the SQL Server hardware
- Adding more processors



Provides an overview of SQL Server workload and can be used for troubleshooting CPU bottlenecks.

CPU Usage(%) by Process (last hours only) and By Database:

Current Database Object Execution Statistics Report:

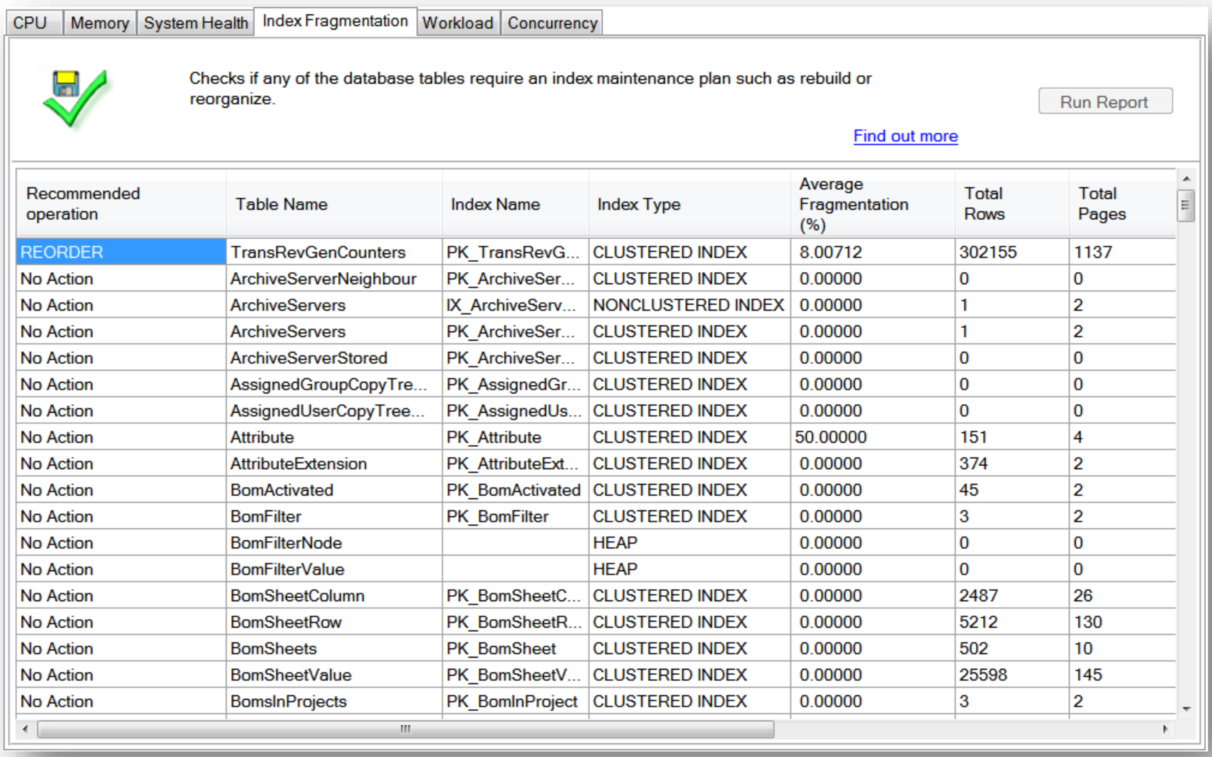
Object Name	Total CPU time (%)	Total CPU time(ms)	Avg. CPU time(ms)	Avg Plan Execution count	Min Plan Execution count
Msg_CheckMsg	45.35	18817.94	2.26	8316.00	8316.00
GenerateNotificationMsgs	22.68	9410.23	2.22	4240.00	4240.00
GetBroadcastEvents	8.57	3556.13	0.14	25441.00	25441.00
ReportBroadcastEventOnNewMessage	6.95	2883.08	0.68	4243.00	4243.00
ReportItemBroadcastEventOnFileFolderChange	6.39	2650.15	0.62	4243.00	4243.00
Arc_GetAllServers	3.26	1354.28	0.05	25441.00	25441.00

i) Index Fragmentation

SQL Server uses indexes to provide fast access to information at the request of users or applications. As the table data grows or changes, the Database Engine maintains these indexes. Over time, the indexes can experience fragmentation, especially in databases like SOLDIWORKS PDM in which there is a lot of insert, update, and delete activity. Fragmentation occurs in an index when the physical ordering on the 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 performance of the application. It can also cause the use of more disk space than is actually necessary. You can correct index fragmentation by reorganizing or rebuilding the index.

The easiest way to check for index fragmentation is to run **Status Report** > **Performance** > **Index Fragmentation**.

The **Recommended operation** column in the report states if a specific table or table index requires a specific maintenance task.



Checks if any of the database tables require an index maintenance plan such as rebuild or reorganize.

Recommended operation	Table Name	Index Name	Index Type	Average Fragmentation (%)	Total Rows	Total Pages
REORDER	TransRevGenCounters	PK_TransRevG...	CLUSTERED INDEX	8.00712	302155	1137
No Action	ArchiveServerNeighbour	PK_ArchiveSer...	CLUSTERED INDEX	0.00000	0	0
No Action	ArchiveServers	IX_ArchiveServ...	NONCLUSTERED INDEX	0.00000	1	2
No Action	ArchiveServers	PK_ArchiveSer...	CLUSTERED INDEX	0.00000	1	2
No Action	ArchiveServerStored	PK_ArchiveSer...	CLUSTERED INDEX	0.00000	0	0
No Action	AssignedGroupCopyTre...	PK_AssignedGr...	CLUSTERED INDEX	0.00000	0	0
No Action	AssignedUserCopyTree...	PK_AssignedUs...	CLUSTERED INDEX	0.00000	0	0
No Action	Attribute	PK_Attribute	CLUSTERED INDEX	50.00000	151	4
No Action	AttributeExtension	PK_AttributeExt...	CLUSTERED INDEX	0.00000	374	2
No Action	BomActivated	PK_BomActivated	CLUSTERED INDEX	0.00000	45	2
No Action	BomFilter	PK_BomFilter	CLUSTERED INDEX	0.00000	3	2
No Action	BomFilterNode		HEAP	0.00000	0	0
No Action	BomFilterValue		HEAP	0.00000	0	0
No Action	BomSheetColumn	PK_BomSheetC...	CLUSTERED INDEX	0.00000	2487	26
No Action	BomSheetRow	PK_BomSheetR...	CLUSTERED INDEX	0.00000	5212	130
No Action	BomSheets	PK_BomSheet	CLUSTERED INDEX	0.00000	502	10
No Action	BomSheetValue	PK_BomSheetV...	CLUSTERED INDEX	0.00000	25598	145
No Action	BomsInProjects	PK_BomInProject	CLUSTERED INDEX	0.00000	3	2

Reorganizing an index does not block user access to the index while the reorganization is in process. However, rebuilding or recreating the index does prevent user access to the index. The exception to this is when using the ***ALTER INDEX REBUILD*** statement with the **ONLINE** = **ON** option.

Periodically checking for index fragmentation and taking any necessary corrective action is important to maintaining the performance of your SOLIDWORKS PDM system. The rate at which fragmentation occurs depends on user activity. In general, SOLIDWORKS Technical Support recommends checking index fragmentation at least monthly.

Use the following values to determine the best method to remove the fragmentation:

- Reorganize the index when index fragmentation falls between 5% to 30%.
- Rebuild the index when index fragmentation is greater than 30%.

For information about how to establish a maintenance plan that rebuilds the indexes for a SOLDIWORKS PDM database, see the following Knowledge Base solutions:

- S-042552 *Using SOLIDWORKS® PDM, what is the recommended procedure to rebuild the file vault databases using a maintenance plan in Microsoft® SQL server?*

- *S-074949 In SOLIDWORKS® PDM, how should I set up a custom script to reorganize or rebuild only those indexes that experience fragmentation?*

The purpose of the indexes is to improve the speed of reading data from the database. However, it is possible to degrade the overall performance with a poor index design because of the impact on database writing. You should not attempt to create or modify indexes manually because this can lead to performance problems or even data corruption.

j) Tempdb Files

Aside from the performance bottleneck issues discussed earlier, the only problem that is likely to occur with tempdb is that disk space runs out. For information about how to handle this situation, see the following articles on the Microsoft website:

- tempdb Performance:

  https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15

- How to shrink the tempdb database in SQL Server:

  http://support.microsoft.com/kb/307487

You can avoid resizing data files by checking the size of tempdb occasionally.

k) Transaction Log file

If log records were never deleted from the transaction log, the log would eventually consume all of the disk space that is available to the physical log files. Under the Simple Recovery model, log truncation automatically frees space for reuse by the transaction log, after a backup checkpoint. Therefore, this should not become a problem if you make regular backups.

If you are using the Full Recovery option, other factors can influence log truncation, so pay extra attention to log file growth.

The only time virtual log files affect system performance is if the log files are defined by small **size** and **growth_increment** values. If these log files grow to a large size because of many small increments, there will be many virtual log files. This can slow down database startup and the backup and restore operations. The recommendation is to assign log files a **size** value close to the final size you require and to assign a relatively large **growth_increment** value.

- The Transaction Log (SQL Server):

  https://docs.microsoft.com/en-us/sql/relational-databases/logs/the-transaction-log-sql-server?view=sql-server-ver15

5) References

There are many sources for technical information about these topics including the following. This is by no means an exhaustive list.

a) SOLIDWORKS PDM Solutions

Many SOLIDWORKS solutions address specific aspects of SQL Server performance. Here is a list of some of these solutions. Because SOLIDWORKS Technical Support adds new solutions to the Knowledge Base regularly, be sure to consult the Knowledge Base whenever you encounter a problem.

Solution	Title
S-072960	Using SOLIDWORKS® PDM 2017, what are the recommended steps to set up and configure Microsoft® SQL Server Enterprise edition for database replication?
S-069784	Is there documentation about how to set up a server side trace on the Microsoft® SQL Server 2005 or Microsoft SQL Server 2008 database software to troubleshoot SOLIDWORKS® PDM blocking chains and deadlocks?
S-068611	What SQL Server Extended Events sessions are recommended to be used for troubleshooting SOLIDWORKS PDM?
S-056773	When using a storage area network (SAN) what could cause SOLIDWORKS® PDM performance issues?
S-00581	Are there any network settings that users should be on the lookout for when experiencing PDMWorks performance issues?
S-057205	What is an SQL Server Profiler trace and how is it used to monitor SQL performance against SOLIDWORKS® PDM databases?
S-025698	Which counters should be used in PerfMon to conduct a performance baseline of Microsoft® SQL Server?
S-027967	Is there a way to set up performance email alerts to help monitor a SOLIDWORKS® PDM file vault database in Microsoft® SQL Server?
S-037734	How is SOLIDWORKS® PDM performance improved when working remotely through a VPN or WAN with limited bandwidth?
S-055284	What can affect search performance in a SOLIDWORKS® PDM vault database?
S-042552	Using SOLIDWORKS® PDM, what is the recommended procedure to rebuild the file vault databases using a maintenance plan in Microsoft® SQL server?
S-055090	What are some best practices to manage memory to avoid Microsoft® SQL Server process hosting SOLIDWORKS® PDM databases from taking all available operating system memory?
S-027967	Is there a way to set up performance email alerts to help monitor a SOLIDWORKS® PDM file vault database in Microsoft® SQL Server?
S-07794	How can I prevent the Microsoft® SQL database transaction log file ('.LDF') for a SOLIDWORKS® PDM database from growing unexpectedly in size and how do I shrink a large log file size?

S-029263	What is the difference between Simple Recovery and Full Recovery for the Microsoft® SQL Server Recovery Model option?

b) Microsoft Knowledge Base

Microsoft provides a vast amount of information about SQL Server performance. The following articles are some of the most useful and relevant to SOLIDWORKS PDM installations.

Title	URL
Monitor Resource Usage (System Monitor)	https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/monitor-resource-usage-system-monitor?view=sql-server-ver15
Performance Analysis of Logs (PAL) Tool	https://github.com/clinthuffman/PAL
Reorganize and Rebuilding Indexes	https://docs.microsoft.com/en-us/sql/relational-databases/indexes/reorganize-and-rebuild-indexes?view=sql-server-ver15
TempDB Database	https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15
The Transaction Log (SQL Server)	https://docs.microsoft.com/en-us/sql/relational-databases/logs/the-transaction-log-sql-server?view=sql-server-ver15