DB Analysis & Tuning

# Heading 1

**January 3, 2019**

# **Overview**

The expansive market of Donnelly Financial Services has fashioned a substantial use of software, applications, software services, data, and databases.

To determine whether your SQL Server system is performing optimally, take performance measurements at regular intervals over time, even when no problems occur, to establish a server performance baseline. Compare each new set of measurements with those taken earlier.

The following areas affect the performance of SQL Server:

* System resources (hardware)
* Network architecture
* The operating system
* Database applications
* Client applications

At a minimum, use baseline measurements to determine:

* Peak and off-peak hours of operation.
* Production-query or batch-command response times.
* Database backup and restore completion times.

After you establish a server performance baseline, compare the baseline statistics to current server performance. Numbers far above or far below your baseline are candidates for further investigation. They may indicate areas in need of tuning or reconfiguration. For example, if the amount of time to execute a set of queries increases, examine the queries to determine if they can be rewritten, or if column statistics or new indexes must be added.

# PERFORMANCE PROBLEM IDENTIFICATION

It is often more effective to use several Microsoft SQL Server or Microsoft Windows tools together to isolate database performance problems than to use one tool at a time. For example, the graphical Execution Plan feature, also called Showplan, helps you quickly recognize deadlocks in a single query. However, you can recognize some other performance problems more easily if you use the monitoring features of SQL Server and Windows together.

SQL Server Profiler can be used to monitor and troubleshoot Transact-SQL and application-related problems. System Monitor can be used to monitor hardware and other system-related problems.

You can monitor the following areas to troubleshoot problems:

* SQL Server stored procedures or batches of Transact-SQL statements submitted by user applications.
* User activity, such as blocking locks or deadlocks.
* Hardware activity, such as disk usage.

Problems can include:

* Application development errors involving incorrectly written Transact-SQL statements.
* Hardware errors, such as disk- or network-related errors.
* Excessive blocking due to an incorrectly designed database.

## Tools for Common Performance Problems

Equally important is careful selection of the performance problem that you want each tool to monitor or tune. The tool and the utility depend on the type of performance problem you want to resolve.

The following topics describe a variety of monitoring and tuning tools and the problems they uncover.

## Monitor CPU Usage

Monitor an instance of Microsoft SQL Server periodically to determine whether CPU usage rates are within normal ranges. A continually high rate of CPU usage may indicate the need to upgrade the CPU or add multiple processors. Alternatively, a high CPU usage rate may indicate a poorly tuned or designed application. Optimizing the application can lower CPU utilization.

An efficient way to determine CPU usage is to use the **Processor:% Processor Time** counter in System Monitor. This counter monitors the amount of time the CPU spends executing a thread that is not idle. A consistent state of 80 percent to 90 percent may indicate the need to upgrade your CPU or add more processors. For multiprocessor systems, monitor a separate instance of this counter for each processor. This value represents the sum of processor time on a specific processor. To determine the average for all processors, use the **System: %Total Processor Time** counter instead.

Optionally, you can also monitor the following counters to monitor processor usage:

* **Processor: % Privileged Time**

Corresponds to the percentage of time the processor spends on execution of Microsoft Windows kernel commands, such as processing of SQL Server I/O requests. If this counter is consistently high when the **Physical Disk** counters are high, consider installing a faster or more efficient disk subsystem.

**Note**

Different disk controllers and drivers use different amounts of kernel processing time. Efficient controllers and drivers use less privileged time, leaving more processing time available for user applications, increasing overall throughput.

* **Processor: %User Time**

Corresponds to the percentage of time that the processor spends on executing user processes such as SQL Server.

* **System: Processor Queue Length**

Corresponds to the number of threads waiting for processor time. A processor bottleneck develops when threads of a process require more processor cycles than are available. If more than a few processes attempt to utilize the processor's time, you might need to install a faster processor. Or, if you have a multiprocessor system, you could add a processor.

When you examine processor usage, consider the type of work that the instance of SQL Server performs. If SQL Server performs many calculations, such as queries involving aggregates or memory-bound queries that require no disk I/O, 100 percent of the processor's time can be used. If this causes the performance of other applications to suffer, try changing the workload. For example, dedicate the computer to running the instance of SQL Server.

Usage rates around 100 percent, where many client requests are being processed, may indicate that processes are queuing up, waiting for processor time, and causing a bottleneck. Resolve the problem by adding faster processors.

## Monitor Disk Usage

Microsoft SQL Server uses Microsoft Windows operating system input/output (I/O) calls to perform read and write operations on your disk. SQL Server manages when and how disk I/O is performed, but the Windows operating system performs the underlying I/O operations. The I/O subsystem includes the system bus, disk controller cards, disks, tape drives, CD-ROM drive, and many other I/O devices. Disk I/O is frequently the cause of bottlenecks in a system.

Monitoring disk activity involves two areas of focus:

* Monitoring Disk I/O and Detecting Excess Paging
* Isolating Disk Activity That SQL Server Creates

### Monitoring Disk Usage

Microsoft SQL Server performance depends heavily on the I/O Subsystem (IOS). Latency in the IOS can result in many performance problems. For example, you may experience slow response times and problems caused by tasks timing out. It is critical that you monitor your disk usage.

Disk counters that you can monitor to determine disk activity are divided into the following two groups:

* Primary
  + PhysicalDisk: Avg. Disk sec/Write
  + PhysicalDisk: Avg. Disk sec/Read
* Secondary
  + PhysicalDisk: Avg. Disk Queue Length
  + PhysicalDisk: Disk Bytes/sec
  + PhysicalDisk: Disk Transfers/sec

Each disk should be monitored individually. Note that using the (\_Total) instance can be misleading and mask problem areas. This is because the (\_Total) instance sums and averages all of the disks together.

**NOTE**: If you are using mount points we recommend that you use the Logical Disk object instead of the Physical Disk object. Logical Disk displays the mount point path instead of the physical drive number.

In System Monitor, the **Avg. Disk sec/Write**and **Avg. Disk sec/Read** counters are considered “primary”. These counters should be examined first and do not need any additional information to evaluate the drive performance. These counters determine the average latency of an I/O request.

**Avg. Disk Sec/Read** is the average time in seconds of a read of data from the disk. The following list shows ranges of possible values and what the ranges represent:

* Less than 10 ms - very good
* Between 10 - 20 ms - okay
* Between 20 - 50 ms - slow, needs attention
* Greater than 50 ms – Serious I/O bottleneck

**Avg. Disk Sec/Write** is the average time in seconds of a write of data to the disk. The guidelines for the Avg. Disk Sec/Read values apply here.

**NOTE**: The numbers listed in this section are for general reference. If you have a very high requirement of application response time on a busy system, meeting the disk response time with these numbers might not be sufficient.

If all or most of the drives report high latency, the bottleneck is probably in the communication medium (such as SAN HBA, Switches, Fiber, Frontend Adaptor CPUs, and Cache). If only one drive or a select few report latency, the bottleneck will generally be in the JBOD (number of disks). To further examine this, review the secondary counters for the drives reporting high latency. If all drives are below their threshold there is no reason to examine the secondary counters.

**NOTE**: In System Monitor, it is important to monitor using the maximum field. Using the average field in System Monitor can be misleading.

You should only use the secondary counters for the drive(s) that have high latency. If the drive has acceptable latency, there is no point in moving forward. **Disk Bytes/sec** and **Disk Transfers/sec** are used to determine the size of and number of I/O requests. These counters can help determine if the number of disks or the communication medium is the source of the latency. You can also use **Avg. Disk Queue** Length to validate the communication medium. Generally, a value greater than 32 represents a bottleneck that can increase latency.

**Disk Transfers/sec** is comprised of both **Disk Reads/sec** and **Disk Writes/sec**. You can use these counters to determine if the drive does not have enough supporting disks. When you use these counters you might need to adjust the values for the type of RAID you have implemented. To determine what values to use, use the following formulas:

* Raid 0 -- I/Os per disk = (reads + writes) / number of disks
* Raid 1 -- I/Os per disk = [reads + (2 \* writes)] / 2
* Raid 5 -- I/Os per disk = [reads + (4 \* writes)] / number of disks
* Raid 10 -- I/Os per disk = [reads + (2 \* writes)] / number of disks

For example, if the maximum value for Disk Transfers/sec is 1800, you could determine that the drive will need at least 10 15k RPM disks in its RAID group. Generally, a 15k RPM disk is capable of approximately 180 I/O requests per second (IOPS). 180\*10 = 1800. For a higher value you might require more than 10 disks.

**NOTE**: Consult your hardware vendor to identify the precise amount of IOPS your disks are capable of handling. The average seek time and rotational latency can affect the IOPS output. All disks are NOT created equal.

If the latency is consistently high, you can determine the root cause by using the secondary counters. If the latency is due to the number of disks, consider the following:

* Use a faster disk drive.
* Move frequently accessed files to a separate disk, server, or SAN.
* Add disks to a RAID array if you are using one.
* Use a faster RAID type, such as RAID 10.
* Stop sharing disks with other volumes or LUNs.

If the latency is due to the communication medium, consider the following:

* Increase the queue depth.
* Move frequently accessed files to a separate disk, server, or SAN.
* Validate the SAN cache.
* Use multiple paths.

The following three questions can be used to help identify if there is a storage bottleneck and where it likely is:

* Is there observed latency? (Avg Disk Sec/Read > 0.020 or Avg Disk Sec/Write > 0.020)
* Is the latency observed on all (most) of the disks (LUNs) or only a single (few) disk (LUN).
  + This question helps us understand if the problem is leaning towards a general lack of communication between the server and the storage or if the issue is more likely due to limitations of the physical spindles.
  + If most of the disks are observed with latency at the same time it may indicate that the latency is due to a communication bottleneck, such as: an HBA, Switch, SAN Port, or SAN CPU.
  + If there are many LUNs from the same storage device and only a single or few are observed with latency, the issue is likely due to the LUN.
* Finally, compare the disk throughput (Disk Transfers/sec & Disk Bytes/sec) during the time that latency was observed to the time when the maximum throughput is observed.
  + If latency always grows in proportion with the throughput, the issue may be with the physical spindles; though, this does not rule out the communication layer. Engage your storage administrator to identify if the physical spindles are capable of handling the throughput observed with Disk Transfers/sec and Disk Bytes/sec.
  + If latency is found to be much lower when the activity is much higher than the bottleneck is likely not due to the physical spindles (JBOD). A storage administrator should be engaged to assist in reviewing the storage fabric (HBA, switches, SAN CPU, Ports, ...).

In addition to these recommendations, consider tuning queries that generate large amounts of I/O. To identify queries that consume large amounts of I/O, use the [SYS.DM\_EXEC\_QUERY\_STATS [Jump](http://msdn.microsoft.com/en-us/library/ms189741.aspx)](http://msdn.microsoft.com/en-us/library/ms189741.aspx)DMV. Thedynamic management views (DMVs) have metrics for both reads and writes and are displayed by the query. You can also include the query plan and SQL command text by joining the [SYS.DM\_EXEC\_SQL\_TEXT [Jump](http://msdn.microsoft.com/en-us/library/ms181929.aspx)](http://msdn.microsoft.com/en-us/library/ms181929.aspx)and [SYS.DM\_EXEC\_QUERY\_PLAN [Jump](http://msdn.microsoft.com/en-us/library/ms189747.aspx)](http://msdn.microsoft.com/en-us/library/ms189747.aspx)dynamic management functions with the CROSS APPLY operator.

Following is a sample query using [SYS.DM\_EXEC\_QUERY\_STATS [Jump](http://msdn.microsoft.com/en-us/library/ms189741.aspx)](http://msdn.microsoft.com/en-us/library/ms189741.aspx):

SELECT TOP 25 execution\_count, plan\_generation\_num, last\_execution\_time,  
        total\_worker\_time, last\_worker\_time, min\_worker\_time, max\_worker\_time,  
        total\_logical\_reads, last\_logical\_reads, min\_logical\_reads,  max\_logical\_reads,  
        total\_physical\_reads, last\_physical\_reads, min\_physical\_reads,  max\_physical\_reads,  
        total\_logical\_writes, last\_logical\_writes, min\_logical\_writes, max\_logical\_writes,  
        total\_elapsed\_time, last\_elapsed\_time, min\_elapsed\_time, max\_elapsed\_time,  
        (SUBSTRING(s2.text,  statement\_start\_offset / 2, ( (CASE WHEN statement\_end\_offset = -1 THEN(LEN(CONVERT(nvarchar(max),s2.text)) \* 2) ELSE statement\_end\_offset END)  - statement\_start\_offset) / 2)  )  ASsql\_statement,   
        text, p.query\_plan  
FROM sys.dm\_exec\_query\_stats qs  
     CROSS APPLY sys.dm\_exec\_sql\_text(qs.plan\_handle) s2  
     CROSS APPLY sys.dm\_exec\_query\_plan(qs.plan\_handle) P  
ORDER BY total\_physical\_reads DESC

Additionally, the **PENDING\_DISK\_IO\_COUNT**column in [SYS.DM\_OS\_SCHEDULERS [Jump](http://msdn.microsoft.com/en-us/library/ms177526.aspx)](http://msdn.microsoft.com/en-us/library/ms177526.aspx)can indicate disk subsystem problems. You should further investigate any sustained value for each processor. You can use[SYS.DM\_IO\_PENDING\_IO\_REQUESTS [Jump](http://msdn.microsoft.com/en-us/library/ms188762.aspx)](http://msdn.microsoft.com/en-us/library/ms188762.aspx)to identify what the waiting requests are and associate them to database files. [SYS.DM\_IO\_VIRTUAL\_FILE\_STATS [Jump](http://msdn.microsoft.com/en-us/library/ms190326.aspx)](http://msdn.microsoft.com/en-us/library/ms190326.aspx)reports I/O statistics on both data and log files.

Another resource you can use is the [SYS.DM\_OS\_WAIT\_STATS [Jump](http://msdn.microsoft.com/en-us/library/ms179984.aspx)](http://msdn.microsoft.com/en-us/library/ms179984.aspx)DMV. Use this DMV to determine what the engine is frequently waiting for and target that area for tuning. You might have an IOS bottleneck if PAGEIOLATCH waits represent some of the highest waits. PAGEIOLATCH waits indicate the amount of time the database engine has been waiting for the IOS. PAGEIOLATCH has various modes and more waits on PAGEIOLATCH\_SH indicate a read bottleneck, whereas PAGEIOLATCH\_EX indicates a write bottleneck.

Isolating Disk Activity Created by SQL Server

You can monitor the following counters to determine the amount of I/O generated by SQL Server components:

* SQL Server:Buffer Manager:Page reads/sec
* SQL Server:Buffer Manager:Page writes/sec
* SQL Server:Buffer Manager:Checkpoint pages/sec
* SQL Server:Buffer Manager:Lazy writes/sec

In System Monitor, these counters monitor the amount of I/O generated by SQL Server components by examining the following performance areas:

* Writing pages to disk
* Reading pages from disk

If the values for these counters approach the capacity limit of the hardware I/O subsystem, attempt to reduce the values by tuning your application or database to reduce I/O operations (such as index coverage, better indexes, or normalization), increasing the I/O capacity of the hardware, or adding memory. For example, you can use the Database Engine Tuning Advisor to analyze typical SQL Server workloads and produce recommendations for indexes, indexed views, and partitioning to improve server performance. For more information about Database Engine Tuning Advisor, see [Tuning the Physical Database Design [Jump](http://msdn.microsoft.com/en-us/library/ms191531.aspx)](http://msdn.microsoft.com/en-us/library/ms191531.aspx)

*NOTE: These two counters do not include I/O generated by transaction log flushes. Most of the transaction log I/Os are writes.*

## Monitor Memory Usage

To monitor for a low-memory condition, use the following object counters:

* **Memory: Available Bytes**
* **Memory: Pages/sec**

The **Available Bytes** counter indicates how many bytes of memory are currently available for use by processes. The **Pages/sec** counter indicates the number of pages that either were retrieved from disk due to hard page faults or written to disk to free space in the working set due to page faults.

Low values for the **Available Bytes** counter can indicate that there is an overall shortage of memory on the computer or that an application is not releasing memory. A high rate for the **Pages/sec** counter could indicate excessive paging. Monitor the **Memory: Page Faults/sec** counter to make sure that the disk activity is not caused by paging.

A low rate of paging (and hence page faults) is typical, even if the computer has plenty of available memory. The Microsoft Windows Virtual Memory Manager (VMM) takes pages from SQL Server and other processes as it trims the working-set sizes of those processes. This VMM activity tends to cause page faults. To determine whether SQL Server or another process is the cause of excessive paging, monitor the **Process: Page Faults/sec** counter for the SQL Server process instance.

For more information about resolving excessive paging, see the Windows operating system documentation.

## Isolating Memory Used by SQL Server

By default, SQL Server changes its memory requirements dynamically, on the basis of available system resources. If SQL Server needs more memory, it queries the operating system to determine whether free physical memory is available and uses the available memory. If SQL Server does not need the memory currently allocated to it, it releases the memory to the operating system. However, you can override the option to dynamically use memory by using the **minservermemory**, and **maxservermemory** server configuration options. For more information, see [Server Memory Options](https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/server-memory-server-configuration-options?view=sql-server-2017).

To monitor the amount of memory that SQL Server uses, examine the following performance counters:

* **Process: Working Set**
* **SQL Server: Buffer Manager: Buffer Cache Hit Ratio**
* **SQL Server: Buffer Manager: Database Pages**
* **SQL Server: Memory Manager: Total Server Memory (KB)**

The **WorkingSet** counter shows the amount of memory that is used by a process. If this number is consistently below the amount of memory that is set by the **min server memory** and **max server memory**server options, SQL Server is configured to use too much memory.

The **Buffer Cache Hit Ratio** counter is specific to an application. However, a rate of 90 percent or higher is desirable. Add more memory until the value is consistently greater than 90 percent. A value greater than 90 percent indicates that more than 90 percent of all requests for data were satisfied from the data cache.

If the **TotalServerMemory (KB)** counter is consistently high compared to the amount of physical memory in the computer, it may indicate that more memory is required.

## Determining Current Memory Allocation

The following query returns information about currently allocated memory.

SELECT

(physical\_memory\_in\_use\_kb/1024) AS Memory\_usedby\_Sqlserver\_MB,

(locked\_page\_allocations\_kb/1024) AS Locked\_pages\_used\_Sqlserver\_MB,

(total\_virtual\_address\_space\_kb/1024) AS Total\_VAS\_in\_MB,

process\_physical\_memory\_low,

process\_virtual\_memory\_low

FROM sys.dm\_os\_process\_memory;

## SQL Server, General Statistics Object

The **SQLServer:General Statistics** object in SQL Server provides counters to monitor general server-wide activity, such as the number of current connections and the number of users connecting and disconnecting per second from computers running an instance of SQL Server. This can be useful when you are working on large online transaction processing (OLTP) type systems where there are many clients connecting and disconnecting from an instance of SQL Server.

This table describes the SQL Server **General Statistics** counters.

| **SQL Server General Statistics counters** | **Description** |
| --- | --- |
| **Active Temp Tables** | Number of temporary tables/table variables in use. |
| **Connection resets/sec** | Total number of logins started from the connection pool. |
| **Event Notifications Delayed Drop** | Number of event notifications waiting to be dropped by a system thread. |
| **HTTP Authenticated Requests** | Number of authenticated HTTP requests started per second. |
| **Logical Connections** | Number of logical connections to the system.  The main purpose of logical connections is to service multiple active result sets (MARS) requests. For MARS requests, every time that an application makes a connection to SQL Server, there may be more than one logical connection that corresponds to a physical connection.  When MARS is not used, the ratio between physical and logical connections is 1:1. Therefore, every time that an application makes a connection to SQL Server, logical connections will increase by 1. |
| **Logins/sec** | Total number of logins started per second. This does not include pooled connections. |
| **Logouts/sec** | Total number of logout operations started per second. |
| **Mars Deadlocks** | Number of MARS deadlocks detected. |
| **Non-atomic yield rate** | Number of non-atomic yields per second. |
| **Processes blocked** | Number of currently blocked processes. |
| **SOAP Empty Requests** | Number of empty SOAP requests started per second. |
| **SOAP Method Invocations** | Number of SOAP method invocations started per second. |
| **SOAP Session Initiate Requests** | Number of SOAP Session initiate requests started per second. |
| **SOAP Session Terminate Requests** | Number of SOAP Session terminate requests started per second. |
| **SOAP SQL Requests** | Number of SOAP SQL requests started per second. |
| **SOAP WSDL Requests** | Number of SOAP Web Service Description Language requests started per second. |
| **SQL Trace IO Provider Lock Waits** | Number of waits for the File IO Provider lock per second. |
| **Temp Tables Creation Rate** | Number of temporary tables/table variables created per second. |
| **Temp Tables For Destruction** | Number of temporary tables/table variables waiting to be destroyed by the cleanup system thread. |
| **Tempdb recovery unit id** | Number of duplicate tempdb recovery unit id generated. |
| **Tempdb rowset id** | Number of duplicate tempdb rowset id generated. |
| **Trace Event Notifications Queue** | Number of trace event notification instances waiting in the internal queue to be sent through Service Broker. |
| **Transactions** | Number of transaction enlistments (local, DTC, bound all combined). |
| **User Connections** | Counts the number of users currently connected to SQL Server. |

## Monitor Resource Usage

use the System Monitor graphical tool to measure the performance of SQL Server. You can view SQL Server objects, performance counters, and the behavior of other objects, such as processors, memory, cache, threads, and processes. Each of these objects has an associated set of counters that measure device usage, queue lengths, delays, and other indicators of throughput and internal congestion.

**Note**

System Monitor replaced Performance Monitor after Windows NT 4.0.

## Benefits of System Monitor

System Monitor can be useful to monitor Windows operating system and SQL Server counters at the same time to determine any correlation between the performance of SQL Server and Windows. For example, monitoring the Windows disk input/output (I/O) counters and the SQL Server Buffer Manager counters at the same time can reveal the behavior of the entire system.

System Monitor allows you to obtain statistics on current SQL Server activity and performance. Using System Monitor, you can:

* View data simultaneously from any number of computers.
* View and change charts to reflect current activity, and show counter values that are updated at a frequency that the user defines.
* Export data from charts, logs, alert logs, and reports to spreadsheet or database applications for further manipulation and printing.
* Add system alerts that list an event in the alert log and can notify you by issuing a network alert.
* Run a predefined application the first time or every time a counter value goes over or under a user-defined value.
* Create log files that contain data about various objects from different computers.
* Append to one file selected sections from other existing log files to form a long-term archive.
* View current-activity reports, or create reports from existing log files.
* Save individual chart, alert, log, or report settings, or the entire workspace setup for reuse.

**Note**

System Monitor replaced the Performance Monitor after Windows NT 4.0. You can use either the System Monitor or Performance Monitor to do these tasks.

## System Monitor Performance

When you monitor SQL Server and the Microsoft Windows operating system to investigate performance-related issues, concentrate your initial efforts in three main areas:

* Disk activity
* Processor utilization
* Memory usage

Monitoring a computer on which System Monitor is running can affect computer performance slightly. Therefore, either log the System Monitor data to another disk (or computer) so that it reduces the effect on the computer being monitored, or run System Monitor from a remote computer. Monitor only the counters in which you are interested. If you monitor too many counters, resource usage overhead is added to the monitoring process and affects the performance of the computer that is being monitored.

### System Monitor Tasks

| **Task Description** | **Topic** |
| --- | --- |
| Describes when to use System Monitor and discusses performance overhead when you use System Monitor. | [Run System Monitor](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/run-system-monitor?view=sql-server-2017) |
| Describes how to monitor disk counters to determine disk activity and the amount of I/O generated by their SQL Server components. | [Monitor Disk Usage](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/monitor-disk-usage?view=sql-server-2017) |
| Describes how to monitor an instance of Microsoft SQL Server to determine whether CPU usage rates are within normal ranges. | [Monitor CPU Usage](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/monitor-cpu-usage?view=sql-server-2017) |
| Describes how to monitor an instance of SQL Server to confirm that memory usage is within typical ranges. | [Monitor Memory Usage](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/monitor-memory-usage?view=sql-server-2017) |
| Describes how to create an alert that is raised when a threshold value for a System Monitor counter has been reached. | [Create a SQL Server Database Alert](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/create-a-sql-server-database-alert?view=sql-server-2017) |
| Describes how to you create charts, alerts, logs, and reports to monitor an instance of SQL Server. | [Create Charts, Alerts, Logs, and Reports](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/create-charts-alerts-logs-and-reports?view=sql-server-2017) |
| Lists objects and counters that System Monitor uses to monitor activity in computers running an instance of SQL Server. | [Use SQL Server Objects](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/use-sql-server-objects?view=sql-server-2017) |
| Lists objects and counters that System Monitor uses to monitor In-Memory OLTP activity. | [SQL Server XTP (In-Memory OLTP) Performance Counters](https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/sql-server-xtp-in-memory-oltp-performance-counters?view=sql-server-2017) |

### SQL Server, Locks Object

The **SQLServer:Locks** object in Microsoft SQL Server provides information about SQL Server locks on individual resource types. Locks are held on SQL Server resources, such as rows read or modified during a transaction, to prevent concurrent use of resources by different transactions. For example, if an exclusive (X) lock is held on a row within a table by a transaction, no other transaction can modify that row until the lock is released. Minimizing locks increases concurrency, which can improve performance. Multiple instances of the **Locks** object can be monitored at the same time, with each instance representing a lock on a resource type.

This table describes the SQL Server **Locks** counters.

| **SQL Server Locks counters** | **Description** |
| --- | --- |
| **Average Wait Time (ms)** | Average amount of wait time (in milliseconds) for each lock request that resulted in a wait. |
| **Average Wait Time Base** | For internal use only. |
| **Lock Requests/sec** | Number of new locks and lock conversions per second requested from the lock manager. |
| **Lock Timeouts (timeout > 0)/sec** | Number of lock requests per second that timed out, but excluding requests for NOWAIT locks. |
| **Lock Timeouts/sec** | Number of lock requests per second that timed out, including requests for NOWAIT locks. |
| **Lock Wait Time (ms)** | Total wait time (in milliseconds) for locks in the last second. |
| **Lock Waits/sec** | Number of lock requests per second that required the caller to wait. |
| **Number of Deadlocks/sec** | Number of lock requests per second that resulted in a deadlock. |

SQL Server can lock these resources.

| **Item** | **Description** |
| --- | --- |
| **\_Total** | Information for all locks. |
| **AllocUnit** | A lock on an allocation unit. |
| **Application** | A lock on an application-specified resource. |
| **Database** | A lock on a database, including all objects in the database. |
| **Extent** | A lock on a contiguous group of 8 pages. |
| **File** | A lock on a database file. |
| **Heap/BTree** | Heap or BTree (HOBT). A lock on a heap of data pages, or on the BTree structure of an index. |
| **Key** | A lock on a row in an index. |
| **Metadata** | A lock on a piece of catalog information, also called metadata. |
| **Object** | A lock on table, stored procedure, view, etc, including all data and indexes. The object can be anything that has an entry in **sys.all\_objects**. |
| **Page** | A lock on an 8-kilobyte (KB) page in a database. |
| **RID** | Row ID. A lock on a single row in a heap. |

# DMVs: Determine Usage Statistics and Performance of Views

Here is a simple methodology and a couple of scripts used to get information about the **performance of queries that use Views**. The intention of these scripts is to provide indicators of use and performance of various Views found in a database.

## sys.dm\_exec\_query\_optimizer\_info

The DMV [sys.dm\_exec\_query\_optimizer\_info](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-query-optimizer-info-transact-sql?view=sql-server-2017) exposes statistics about the optimizations performed by the SQL Server query optimizer. These values are cumulative and begin recording when SQL Server starts. For more information on the query optimizer, see the [Query Processing Architecture Guide](https://docs.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-2017).

The below common\_table\_expression (CTE) uses this DMV to provide information about the workload, such as the percentage of queries that reference a view. The results returned by this query do not indicate a performance problem by themselves, but can expose underlying issues when combined with users' complaints of slow-performing queries.

SQLCopy

WITH CTE\_QO AS

(

SELECT

occurrence

FROM

sys.dm\_exec\_query\_optimizer\_info

WHERE

([counter] = 'optimizations')

),

QOInfo AS

(

SELECT

[counter]

,[%] = CAST((occurrence \* 100.00)/(SELECT occurrence FROM CTE\_QO) AS DECIMAL(5, 2))

FROM

sys.dm\_exec\_query\_optimizer\_info

WHERE

[counter] IN ('optimizations'

,'trivial plan'

,'no plan'

,'search 0'

,'search 1'

,'search 2'

,'timeout'

,'memory limit exceeded'

,'insert stmt'

,'delete stmt'

,'update stmt'

,'merge stmt'

,'contains subquery'

,'view reference'

,'remote query'

,'dynamic cursor request'

,'fast forward cursor request'

)

)

SELECT

[optimizations] AS [optimizations %]

,[trivial plan] AS [trivial plan %]

,[no plan] AS [no plan %]

,[search 0] AS [search 0 %]

,[search 1] AS [search 1 %]

,[search 2] AS [search 2 %]

,[timeout] AS [timeout %]

,[memory limit exceeded] AS [memory limit exceeded %]

,[insert stmt] AS [insert stmt %]

,[delete stmt] AS [delete stmt]

,[update stmt] AS [update stmt]

,[merge stmt] AS [merge stmt]

,[contains subquery] AS [contains subquery %]

,[view reference] AS [view reference %]

,[remote query] AS [remote query %]

,[dynamic cursor request] AS [dynamic cursor request %]

,[fast forward cursor request] AS [fast forward cursor request %]

FROM

QOInfo

PIVOT (MAX([%]) FOR [counter]

IN ([optimizations]

,[trivial plan]

,[no plan]

,[search 0]

,[search 1]

,[search 2]

,[timeout]

,[memory limit exceeded]

,[insert stmt]

,[delete stmt]

,[update stmt]

,[merge stmt]

,[contains subquery]

,[view reference]

,[remote query]

,[dynamic cursor request]

,[fast forward cursor request])) AS p;

GO

Combine the results of this query with the results of the system view [sys.views](https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-views-transact-sql?view=sql-server-2017) to identify query statistics, query text, and the cached execution plan.

## sys.views

The below CTE provides information about the number of executions, total run time, and pages read from memory. The results can be used to identify queries that may be candidates for optimization.

**Note**

The results of this query can vary depending on the version of SQL Server.

SQLCopy

WITH CTE\_VW\_STATS AS

(

SELECT

SCHEMA\_NAME(vw.schema\_id) AS schemaname

,vw.name AS viewname

,vw.object\_id AS viewid

FROM

sys.views AS vw

WHERE

(vw.is\_ms\_shipped = 0)

INTERSECT

SELECT

SCHEMA\_NAME(o.schema\_id) AS schemaname

,o.Name AS name

,st.objectid AS viewid

FROM

sys.dm\_exec\_cached\_plans cp

CROSS APPLY

sys.dm\_exec\_sql\_text(cp.plan\_handle) st

INNER JOIN

sys.objects o ON st.[objectid] = o.[object\_id]

WHERE

st.dbid = DB\_ID()

)

SELECT

vw.schemaname

,vw.viewname

,vw.viewid

,DB\_NAME(t.databaseid) AS databasename

,t.databaseid

,t.\*

FROM

CTE\_VW\_STATS AS vw

CROSS APPLY

(

SELECT

st.dbid AS databaseid

,st.text

,qp.query\_plan

,qs.\*

FROM

sys.dm\_exec\_query\_stats AS qs

CROSS APPLY

sys.dm\_exec\_sql\_text(qs.plan\_handle) AS st

CROSS APPLY

sys.dm\_exec\_query\_plan(qs.plan\_handle) AS qp

WHERE

(CHARINDEX(vw.schemaname, st.text, 1) > 0)

AND (st.dbid = DB\_ID())

) AS t;

GO

## sys.dmv\_exec\_cached\_plans

The final query provides information about unused views by using the DMV [sys.dmv\_exec\_cached\_plans](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-cached-plans-transact-sql?view=sql-server-2017). However, the execution plan cache is dynamic, and results can vary. As such, use this query over time to determine whether or not a view is actually being used or not.

SQLCopy

SELECT

SCHEMA\_NAME(vw.schema\_id) AS schemaname

,vw.name AS name

,vw.object\_id AS viewid

FROM

sys.views AS vw

WHERE

(vw.is\_ms\_shipped = 0)

EXCEPT

SELECT

SCHEMA\_NAME(o.schema\_id) AS schemaname

,o.name AS name

,st.objectid AS viewid

FROM

sys.dm\_exec\_cached\_plans cp

CROSS APPLY

sys.dm\_exec\_sql\_text(cp.plan\_handle) st

INNER JOIN

sys.objects o ON st.[objectid] = o.[object\_id]

WHERE

st.dbid = DB\_ID();

GO

## DMVs by Category

Dynamic management views and functions have been organized into the following categories.

|  |  |
| --- | --- |
| [Always On Availability Groups Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/always-on-availability-groups-dynamic-management-views-functions?view=sql-server-2017) | [Memory-Optimized Table Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/memory-optimized-table-dynamic-management-views-transact-sql?view=sql-server-2017) |
| [Change Data Capture Related Dynamic Management Views (Transact-SQL)](https://msdn.microsoft.com/library/2a771d7d-693a-4f56-9227-02cd00e0e200) | [Object Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/object-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) |
| [Change Tracking Related Dynamic Management Views](https://msdn.microsoft.com/library/dc8a0af9-fcd8-4c34-9453-5132717c9bdb) | [Query Notifications Related Dynamic Management Views (Transact-SQL)](https://msdn.microsoft.com/library/92eb22d8-33f3-4c17-b32e-e23acdfbd8f4) |
| [Common Language Runtime Related Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/common-language-runtime-related-dynamic-management-views-transact-sql?view=sql-server-2017) | [Replication Related Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/replication-related-dynamic-management-views-transact-sql?view=sql-server-2017) |
| [Database Mirroring Related Dynamic Management Views (Transact-SQL)](https://msdn.microsoft.com/library/04fb21de-1b5e-4a8e-9ca6-1b78ad278db1) | [Resource Governor Related Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/resource-governor-related-dynamic-management-views-transact-sql?view=sql-server-2017) |
| [Database Related Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/database-related-dynamic-management-views-transact-sql?view=sql-server-2017) | [Security-Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/security-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) |
| [Execution Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/execution-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) | [Server-Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/server-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) |
| [Extended Events Dynamic Management Views](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/extended-events-dynamic-management-views?view=sql-server-2017) | [Service Broker Related Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/service-broker-related-dynamic-management-views-transact-sql?view=sql-server-2017) |
| [Filestream and FileTable Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/filestream-and-filetable-dynamic-management-views-transact-sql?view=sql-server-2017) | [Spatial Data Related Dynamic Management Views and Functions (Transact-SQL)](https://msdn.microsoft.com/library/c542ac38-451f-43a5-bf8c-4edd38bb738e) |
| [Full-Text Search and Semantic Search Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/full-text-and-semantic-search-dynamic-management-views-functions?view=sql-server-2017) | [SQL Data Warehouse and Parallel Data Warehouse Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sql-and-parallel-data-warehouse-dynamic-management-views?view=sql-server-2017) |
| [Geo-Replication Dynamic Management Views and Functions (Azure SQL Database)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/geo-replication-dynamic-management-views-and-functions-azure-sql-database?view=sql-server-2017) | [SQL Server Operating System Related Dynamic Management Views (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sql-server-operating-system-related-dynamic-management-views-transact-sql?view=sql-server-2017) |
| [Index Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/index-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) | [Stretch Database Dynamic Management Views (Transact-SQL)](https://msdn.microsoft.com/library/1193efce-a105-49a9-a8b8-26b063485567) |
| [I O Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/i-o-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) | [Transaction Related Dynamic Management Views and Functions (Transact-SQL)](https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/transaction-related-dynamic-management-views-and-functions-transact-sql?view=sql-server-2017) |

# **Goals**

1. **Lorem ipsum dolor sit amet:** Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan.
2. **Sed diam nonummy nibh euismod:** Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

# **Specifications**

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

# **Milestones**

## **Lorem ipsum**

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

## **Dolor sit amet**