DZone® A DEVADA MEDIA PROPERTY

BROUGHT TO YOU IN PARTNERSHIP WITH redgate

Database Monitoring

CONTENTS

WRITTEN BY MATT HILBERT
TECHNOLOGY WRITER AT REDGATE SOFTWARE

The Changing Face of Database Monitoring

Database monitoring has always been central to the role of Database Administrators (DBAs) to minimize downtime and optimize performance. Scheduled monitoring keeps an eye on resource usage like CPU, disk space, memory, and I/O capacity to spot trends and understand when more capacity will be needed. The same information also enables baselines to be established so that, for example, it is immediately apparent if a high-resource utilization is an abnormal spike, a worrying recent trend, or just normal behavior for the period in question.

Alongside scheduled monitoring, reactive monitoring also has its place in responding to alerts about a drop in performance or an increase in deadlocks, and drilling down to the cause of the problem before it becomes an issue. Here, the history and timelines which monitoring provides will help to identify if a stress phenomenon coincides with a particular type of processing, such as a weekly aggregation or a scheduled data import.

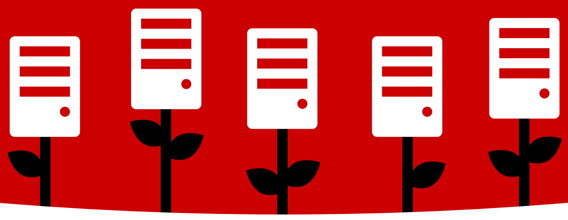
All of this will be familiar to any DBA, but the way databases are developed and managed has seen a big change over the last few years, driven by the adoption of DevOps, the move from on-premises to the cloud, and increasing concerns about data privacy. These three factors are introducing new challenges to database monitoring that every DBA needs to be aware of.

DevOps, for example, encourages the frequent release of small changes to applications, and those changes often mean the database needs to be updated as well. This was highlighted in Redgate's 2018 State of SQL Server Monitoring report which revealed that almost 50% of respondents deploy database changes multiple times a week. That's a big change from a few years ago when deployments were often major events, regarded with trepidation and planned carefully to allow for the expected downtime failed deployments would cause.

The cloud has also become mainstream, and the same report showed that 48% of respondents are already using some cloud technology, and

SQL Monitor

Proactively monitor your growing SQL Server estate



Try it free

redgate

SQL Monitor

Proactively monitor your growing SQL Server estate

- **Alerting** – Discover issues before they have an impact
- **Diagnosis** – Uncover obstacles and find root causes
- **Performance** – See what has the biggest impact on your system
- **Overview** – View your SQL Server estate at a glance
- **Reporting** – Share tailored reports about your servers health

Find out more at
www.red-gate.com/monitor

redgate

moving to the cloud is their biggest concern. Interestingly, SQL Server implementations on Azure and Amazon are the most common cloud technologies used, and this will probably increase now that Azure SQL Managed Instances provide near 100% feature compatibility with on-premises SQL Server, yet also offer the benefits of a cloud service. The task here will be monitoring a mixed server estate, with legacy systems remaining on-premises, and new systems moving to the cloud.

And then there's the privacy question. With new requirements for access to personal data to be restricted, there are demands on DBAs to monitor access rights and privileges — and demonstrate that such a system is in place.

Despite all these changes, many of the reasons for monitoring databases and the methods of doing so remain the same. This Refcard focuses on SQL Server databases but the broad principles are the same for any data-base. DBAs and systems administrators should make the most of built-in tools and resources but also call on third-party tools where necessary to manage their own server estate in the way their business demands.

The Top Five Metrics to Monitor:

Being a DBA is much more than knowing how to install a server and set up a database. One of the most important responsibilities is being proactive by monitoring instances to identify potential problems. But what should be monitored and why? Here are the top five things to monitor in the new era where DevOps, the cloud, and data privacy have entered the picture.

#1 PERFORMANCE

Monitoring database performance has traditionally been about taking baselines, watching resource utilization (CPU, memory, I/O) changes over time, and determining the top ten or so worst performing queries so you can tune them.

With the database increasingly expected to be included in DevOps, leading to more changes, more often, another added measure has come into play. However thorough the testing regime is in develop-ment, a change can reach production which slows down performance, so monitoring the effect frequent changes have, at the time they are deployed, also needs to be added to the list.

#2 SECURITY

The introduction of the GDPR, as well as upcoming legislation like New York's Stop Hacks and Improve Electronic Data Security Act (SHIELD) and the California Consumer Privacy Act (CCPA) have put the spotlight firmly on security. Indeed, the SQL Server Monitoring report mentioned in the introduction revealed that security and protection are the biggest challenge for those responsible for managing SQL Server estates.

So, as well as keeping track of failed logins and how many accounts are in the sysadmin group, DBAs also need to be monitoring for SQL injection attacks, changes in server and database settings, and modifi-cations to permissions, users, and roles.

#3 BACKUPS

There are exceptions, but just about every database should be backed up on a regular basis, including frequent transaction log backups. Un-less you have a job in place to back up every database on an instance by default, it's easy to miss adding new databases. By the way, make sure you have a process in place to test backup files as well.

#4 FILE GROWTH

Over time, database files can run out of free space as can the volumes where these files live. Transactions must consequently wait while data-base files grow, and applications will grind to a halt if there is no more space in the files or space runs out on the volume. While the cloud offers elasticity and the ability to grow volumes almost instantly, this is still the case for on-premises servers.

#5 JOB OUTCOMES

Great DBAs automate everything they can and use SQL Server Agent or some other job scheduler to run the scripts. They understand what the jobs do and the consequences of a job failure or long running job. They also have every job documented so that they can take a day off once in a while.

Choosing Your Monitoring Solution

However, you choose to monitor your SQL Server systems, you'll always need the following elements:

- **A graphical dashboard**, showing the history, timelines, and baselines for your performance metrics. This will allow you to see the general symptoms of any problem and form a hypothe-sis as to the cause.
- **A highly configurable alerting system** which will warn the team immediately of abnormal conditions, errors, possibly security issues and so on.
- **Fast drilldown to the detailed diagnostic data** to check your hypothesis and find out what really happened. I'll cover a variety of built-in tools that can provide this data. If you're using a third-party tool, then you'll often find that lightweight built-in frameworks such as Extended Events will provide the data that underpins the metrics and alerts it provides, so you'll have direct access to the detailed diagnostic data collected around the time a problem occurred.

Over the coming sections, I'll try to help inform your choices, by reviewing the main monitoring capabilities built into SQL Server, the graphical tools provided with SQL Server Management Studio, and finally, third-party monitoring solutions.

SQL Server's Built-in Monitoring Resources

SQL Server and SQL Server Management Studio come with a number of built-in resources that can help with monitoring. It's worthwhile looking into them because they will help you understand what you can do out-of-the-box without needing to invest in a paid-for tool. Similarly, it will reveal any limitations and tell you what you should be looking for in a paid-for tool.

SQL SERVER AGENT

Many alerts and administrative tasks, or jobs, in SQL Server can be scheduled and automated using SQL Server Agent, which runs in the background as a Windows service and can significantly reduce the workload of a DBA.

Alerts can be configured for general SQL Server events, such as when a specific error number occurs; for performance events, such as wait statistics exceeding a certain value; and for Windows Management Instrumentation (WMI) events, such as a new file appearing in a specific folder. The response to the alerts can also be configured, which can be to run a SQL Server Agent job, or send a notification to one or more administrators.

SQL Server Agent jobs contain one or more steps and can be set up to run tasks like backing up a database at a specific time with a set frequency, or whenever a new database is created or attached.

SQL SERVER ERROR LOG

The SQL Server error log contains certain user-defined and system events that can aid with troubleshooting and ensuring processes like backup and restore, batch commands and other scripts have completed successfully.

This can be helpful in detecting current or potential problem areas, including automatic recovery messages (particularly if an instance of SQL Server has been stopped and restarted), kernel messages, or other server-level error messages.

A new error log is created each time an instance of SQL Server is started, and backups of the previous six logs are also retained. The logs are text files and can be viewed with any text editor.

By default, they're located in the `Program Files\Microsoft SQL Server\MSSQL.n\MSSQL\LOG` folder, with the current error log having no suffix and the backups having a sequentially numbered suffix.

The logs can also be viewed in SQL Server Management Studio by expanding the Management folder in Object Explorer, and then opening the SQL Server Logs folder.

PERFORMANCE MONITOR

Performance Monitor, or PerfMon, is a popular tool for general server resource monitoring. It provides a range of counters for monitoring memory, disk I/O, CPU and network usage on a server, and correlates them with the performance counters maintained by SQL Server in a real-time graph.

DBAs typically select the counters and resource metrics they want to track and set up PerfMon to record them at regular intervals. The data can then be imported into Excel or a similar tool for later analysis.

While it's easy to set up and access, (you simply type `perfmon` in the Run window from the Start Menu), it has limited reporting options and it can't access SQL Server internals like Dynamic Management Views (DMVs) and stored procedures.

DYNAMIC MANAGEMENT VIEWS

Under the covers, SQL Server tracks the performance of database sessions and transactions, and makes the information available using DMVs. DMVs can return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance.

A common diagnosis for slow query performance, for example, is a need for more CPU. What if, however, it is down to a high number of I/O waits, caused by a resource, queue or external issue? In this case, the `sys.dm_os_wait_stats` DMV query can be used.

By itself, it will return an unordered table which is hard to understand. Add a little more T-SQL as shown in the following example and the table will be ordered with the highest waits at the beginning and exclude those for which wait times are not required.

```
SELECT wait_type ,
    SUM(wait_time_ms / 1000) AS [wait_time_s]
FROM sys.dm_os_wait_stats DOWS
WHERE wait_type NOT IN ( ‘SLEEP_TASK’, ‘BROKER_TASK_
                       STOP’,
                       ‘SQLTRACE_BUFFER_FLUSH’, ‘CLR_
                       AUTO_EVENT’,
                       ‘CLR_MANUAL_EVENT’, ‘LAZYWRITER_
                       SLEEP’ )
GROUP BY wait_type
ORDER BY SUM(wait_time_ms) DESC
```

This will return a list of the wait types, and the wait times associated with each, so that the real cause can be properly investigated and diagnosed.

DMVs can also be used to check performance counters and see if SQL Server has any memory issues. The top three counters to check are page life expectancy, the number of requests that have to wait for a free page (free list stalls/sec), and the page reads per second. The query to retrieve these values would be:

```
SELECT object_name, counter_name, cntr_value
FROM sys.dm_os_performance_counters
WHERE [object_name] LIKE ‘%Buffer Manager%’
AND [counter_name] in (‘Page life expectancy’,’Free list
stalls/sec’,’Page reads/sec’)
```

These are just two examples of how DMVs give instant access to a wealth of information about everything from wait stats to memory issues, through to finding missing indexes and checking on the status of jobs. The full list of DMVs available is on the Microsoft documentation pages. The only downside is that you need to decide the metrics you want to capture, find the correct DMV to use, and then plan how to use it.

SYSTEM STORED PROCEDURES

Included in the `master` database when SQL Server is installed are around 1,000 system stored procedures to help with admin tasks and provide added metrics out-of-the-box. Just as the names of DMVs are prefixed with sys.dm, so the T-SQL for system-stored procedures always begins with `sp_`, which denotes "special."

Three of the most useful are `sp_helpdb`, `sp_spaceused`, and `sp_who`.

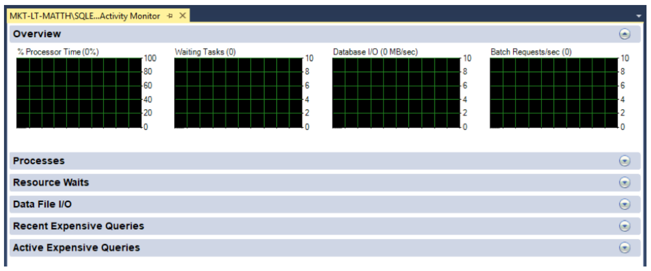
The first returns a list of databases that exist on the server, along with their name, size, owner, date created, and current status; the second shows the name, size, and unallocated space for the currently selected database; and the third lists the active SQL processes, along with any blocking activity, users, and current sessions.

While useful, the information relates to a single server, and is only available as a point-in-time snapshot. Microsoft does, however, provide a handy list of *Database Engine Stored Procedures*, which shows those of most use, when monitoring SQL Server.

SQL Server Management Studio resources

ACTIVITY MONITOR

Included in SSMS is an Activity Monitor which displays various information about what's happening with the instance. It can be accessed by right-clicking on the server in Object Explorer and selecting **Activity Monitor**, by using the key command **Ctrl+Alt+A**, or by selecting the icon in the toolbar:



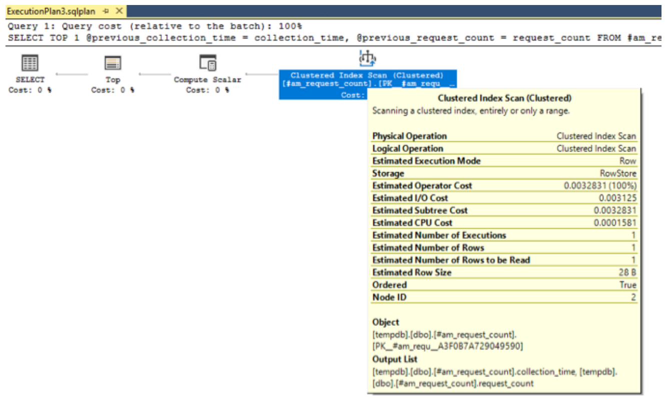
The pane that opens gives a quick overview of activity in four areas:

- ***Processes*** shows information about the current connections to the server
- ***Resource Waits*** shows what SQL Server is waiting on, grouped into categories
- ***Data File I/O*** shows the activity in the database files
- ***Recent Expensive Queries*** and ***Active Expensive Queries*** show expensive queries in the plan cache

While useful for gaining a real-time view of a SQL Server instance, it can only show the activity of one server at a time and leaving it running can be a big drag on performance, particularly if it runs on the same server. As a consequence, it's not suitable as a general tool for long term SQL Server monitoring.

VIEWING EXECUTION PLANS

One handy additional feature Activity Monitor offers is the ability to view how SQL Server executes its query plans and see the resource usage taken up by each step. Simply right-click on the query in the 'Recent Expensive Queries' pane and select **Show Execution Plan**. The diagram that appears shows the computational steps involved in processing the query. Hovering over any of the steps also shows the estimated resource costs involved.



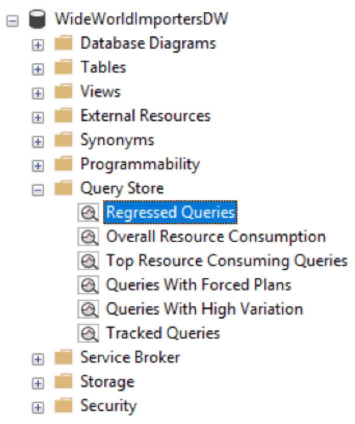
```
Query 1: Query cost (relative to the batch): 100%
SELECT TOP 1 @previous_collection_time = collection_time, @previous_request_count = request_count FROM #am_re
```

Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0032831 (100%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.0032831
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Number of Rows to be Read	1
Estimated Row Size	28 B
Ordered	True
Node ID	2

Object
[tempdb].[dbo].[#am_request_count].[PK__#am_requ__A3F0B7A729049590]
Output List
[tempdb].[dbo].[#am_request_count].collection_time, [tempdb].[dbo].[#am_request_count].request_count



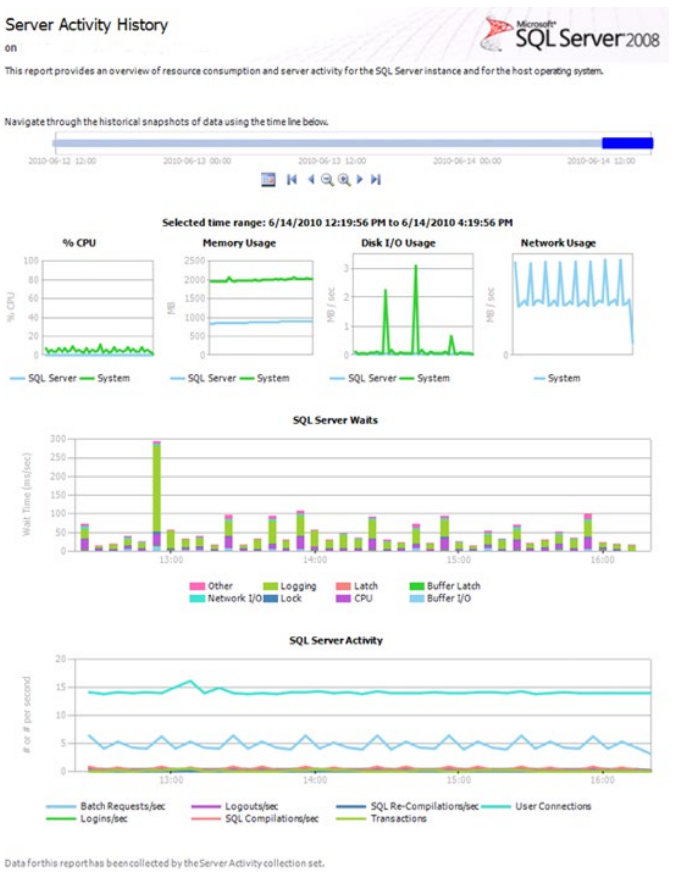
From SQL Server 2016 onwards, those execution plans, together with information about them, are also cached in Query Store. Often referred to as a flight recorder, Query Store provides valuable insights into query plan choices and performance, and simplifies the troubleshooting of performance issues by finding the differences caused by query plan changes. To confirm it's enabled, expand the database in Object Explorer, and the Query Store folder with its built-in reports should appear.

If it's not enabled, it can be turned on and configured by right clicking on the database in Object Explorer, choosing the **Properties** option, and selecting **Query Store**.

SQL DATA COLLECTOR/MANAGEMENT DATA WAREHOUSE

SQL Server Data Collector, together with Management Data Warehouse, is a useful component in SSMS for gathering information centrally about how SQL Server instances are being used and keeping an eye out for problems.

The Data Collector uses a set of SQL Agent jobs to automatically collect performance data from multiple SQL Servers, and the metrics are stored in the data warehouse. It then provides reports on server activity, query statistics and system disk usage, along with suggestions on how performance can be improved. It can also be configured to collect custom data and provide tailor-made reports.

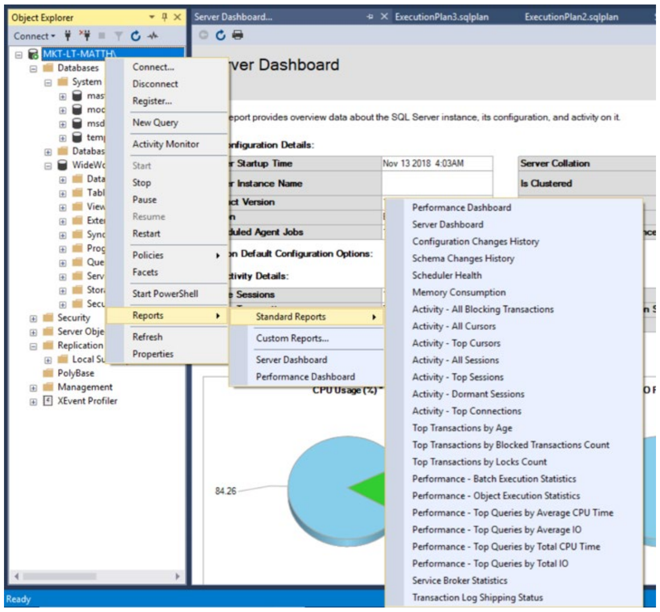


On the plus side, it helps DBAs with tasks like proactive tuning, historical query analysis, performance baselining, growth forecasting, and storage planning. On the downside, it takes time to configure, it's hard to remove once it's up and running, and it should run on a dedicated server of its own to avoid resource issues.

SQL STANDARD REPORTS

Out-of-the-box, SSMS makes a range of reports available at both a server level and database level. The full list of reports can be accessed by right-clicking either the server or database in Object Explorer, and hovering over **Reports** and then **Standard Reports**.

22 server reports cover everything from a history of configuration changes to the transaction log shipping status. A performance dashboard and server dashboard are also accessible from the same list, which provide a useful snapshot of data such as system CPU utilization, current waiting requests, and server configuration and activity.



At the database level, a further 20 reports cover topics like disk usage, data classification, a history of database consistency, and usage statistics.

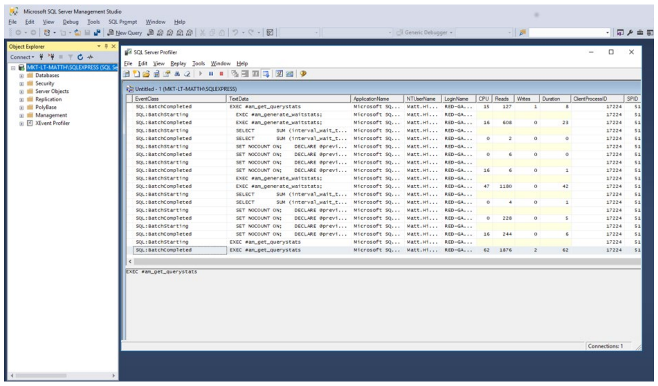
The reports are useful in giving a glimpse into a range of activity, and custom reports can also be created at both the server and database level. They are fixed-in-time, however, and interactivity is limited. They can also be resource intensive, so it would be better to run them during quiet periods.

SQL SERVER PROFILER

For around 20 years, DBAs have used SQL Server Profiler to trace, recreate, and troubleshoot SQL Server problems. A user interface for the SQL Trace utility, it can be opened within SSMS to trace events as they occur in a SQL Server instance. It's typically used to monitor deadlocks and system errors; and identify poor-performing queries so they can be analyzed and tuned.

While robust and useful in showing the health of a SQL Server instance, it has now been deprecated by Microsoft in favor of Extended Events, which require less resources and allow more events to be traced.

Many DBAs still favor it, however, because the easy to navigate GUI provides a wealth of information including CPU time, reads, writes, duration, and the text or stored procedure being called, along with the parameters. It does come at a heavy resource cost, however, and it gathers a lot of information which needs to be sifted through to find the data that's really needed.


Microsoft SQL Server Management Studio
SQL Server Profiler
File Edit View Replay Tools Window Help
Untitled - 1 (MKT-LT-MATTH\SQLEXPRESS)
Connections: 1

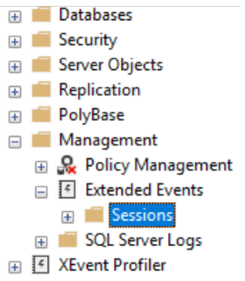
EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID
SQL:BatchCompleted	EXEC #am_get_querystats	Microsoft SQ...	Matt.Hi...	RED-GA...	15	127	1	8	17224	51
SQL:BatchStarting	EXEC #am_generate_waitstats;	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	EXEC #am_generate_waitstats;	Microsoft SQ...	Matt.Hi...	RED-GA...	16	608	0	23	17224	51
SQL:BatchStarting	SELECT SUM (interval_wait_t...	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	SELECT SUM (interval_wait_t...	Microsoft SQ...	Matt.Hi...	RED-GA...	0	2	0	0	17224	51
SQL:BatchStarting	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...	0	6	0	0	17224	51
SQL:BatchStarting	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...	16	6	0	1	17224	51
SQL:BatchStarting	EXEC #am_generate_waitstats;	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	EXEC #am_generate_waitstats;	Microsoft SQ...	Matt.Hi...	RED-GA...	47	1180	0	42	17224	51
SQL:BatchStarting	SELECT SUM (interval_wait_t...	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	SELECT SUM (interval_wait_t...	Microsoft SQ...	Matt.Hi...	RED-GA...	0	4	0	1	17224	51
SQL:BatchStarting	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...	0	228	0	5	17224	51
SQL:BatchStarting	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	SET NOCOUNT ON; DECLARE @previ...	Microsoft SQ...	Matt.Hi...	RED-GA...	16	244	0	6	17224	51
SQL:BatchStarting	EXEC #am_get_querystats	Microsoft SQ...	Matt.Hi...	RED-GA...					17224	51
SQL:BatchCompleted	EXEC #am_get_querystats	Microsoft SQ...	Matt.Hi...	RED-GA...	62	1876	2	62	17224	51

```
EXEC #am_get_querystats
```

SQL SERVER EXTENDED EVENTS

In SQL Server 2016, it was announced that SQL Profiler would be deprecated in future versions and would be replaced by Extended Events. Microsoft was actually late coming to the party with the announcement because it hasn't updated SQL Profiler since 2008. As a result, diagnosing problems for technologies like AlwaysOn Availability Groups, Memory-Optimized Tables, and Azure Storage isn't supported.

The biggest reason for moving to Extended Events to find performance issues and bottlenecks, however, is that it uses very few resources, yet helps to trace and track more events. As of SQL Server 2016, there are 564 events in Extended Events compared to 235 in SQL Server Profiler.



It also has a user-friendly GUI and can be set up easily from within SSMS. Simply expand the Management folder in Object Explorer, right-click on the Sessions folder, and select the **New Session Wizard** or **New Session...** option.

In both cases, the steps to set up a session are straightforward in terms of choosing the events to run and configuring them, etc.

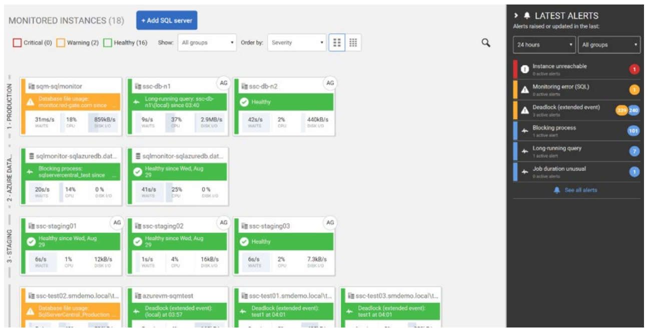
While some DBAs will undoubtedly stay with SQL Server Profiler as long as they can, the ease with which Extended Events can now be set up, and the newer technologies it supports, will attract more and more fans.

The role of third-party monitoring tools

While the built-in monitoring capabilities described previously suit many smaller companies, the use of third-party tools typically increases in line with company size, the number of servers, and the frequency of deployments. The State of SQL Server Monitoring report mentioned earlier revealed that 48% of companies go down the paid-for route, and 38% monitor manually or use a tool built in-house. Worryingly, the remaining 14% either don't monitor their servers or are not sure whether monitoring is in place.

Generally, manual monitoring is more common than paid-for monitoring in companies with under 100 employees, with fewer than ten servers, or when they deploy changes to the database only a few times a year. In such cases, the data required for monitoring can be gathered relatively easily from the built-in functions like PerfMon, Dynamic Management Views (DMVs), Activity Monitor, and Extended Events explained earlier. However, companies often struggle to scale these solutions as the number and size of their database and servers grows.

For larger estates where there are likely to be different versions and editions of SQL Server, as well as instances in the cloud, it will often save time and resources to use a third-party monitoring tool. It doesn't replace the use of DMVs and Extended Events, etc., which will still be required. Instead, it removes the heavy lifting of data collection and management, analyzes the data, and should provide, at the top level, an easy-to-digest picture of activity and any issues and alerts across all your monitored servers:



A typical GUI for a SQL Server monitoring tool, in this case Redgate SQL Monitor.

What to look for in a third-party monitoring tool

At the heart of monitoring any SQL Server estate is the requirement for tailored alerts specific to your particular requirements, baselines to provide a comparison of current performance over past performance, and trends to help with future capacity and infrastructure planning.

A good third-party tool should also provide a customizable at-a-glance overview of an entire SQL Server estate, support hybrid environments, large estates, and Availability Groups; show trends over time; and provide guidance on any performance issues that arise and how to resolve them.

More specifically, if you do decide to look at paid-for tools, there are five factors to consider.

WHAT DOES IT MONITOR AND HOW DOES IT PROVIDE ALERTS?

The first thing you'll want to know is that it provides the coverage you need. Aim for 95%, because the other 5% will come from processes like backups and restores, SLQ Agent jobs, and reporting failures. The solution will need to highlight queries having the biggest impact and feature a focused set of performance metrics, and customizable alerts for the most important operational and performance issues. A nice-to-have is alerts that can be grouped to avoid the alert overload scenario that is common when first introducing a monitoring tool. If you're deploying changes more frequently, look out for a solution that marks on the performance timeline when deployments were made and which database they were made to.

DOES IT PROVIDE BASELINES AND TRENDING INFORMATION?

As we saw earlier, baselines are important because they provide a quick guide to understanding the significance of events like performance spikes and whether they need attention. Trends are also valuable because they can show, for example, the probable point in the future when new resources will be required, which is essential for effective planning.

DOES IT HAVE A GLOBAL OVERVIEW?

As SQL Server estates grow and become more complex, a global overview on one central web-based interface can provide a handy way to check the status of every server in seconds, not hours. It's also worth checking if it offers a way of grouping servers so that you can, for example, group servers with business-critical or sensitive data.

DOES IT OFFER DISTRIBUTED MONITORING?

Connected to the growth of estates is the changing nature of those estates, with servers in different data centers, or on different networks with distinct security protocols. Any third-party monitoring tool should be able to handle this, and also be able to monitor on-premises servers, servers running on VMware, and Azure-based servers at the same time, on the same screen.

DOES IT POSE A RISK TO PERFORMANCE?

A significant problem with legacy and homegrown database monitoring is the performance overhead. They often execute complex data collection queries, set trace flags to capture "verbose output," request "specialist" metrics that are complex to interpret, and so on. This can lead to the "observer effect," where actions required to collect the monitoring data degrade the performance of the monitored server.

A good third-party monitoring tool should limit the data collection to lightweight, efficient SQL operations, exploiting lightweight frameworks such as Extended Events. The installation should also not require agents on each monitored SQL Server instance. This minimizes their exposed "surface area" and reduces risk, and all data processing is done on a separate server. Finally, it should be easy to view the actions taken by the tool itself, to capture the monitoring data.

Written by **Matt Hilbert,** Technology Writer at Redgate Software
Matt Hilbert has 20 years' experience working for lots of the world's biggest tech companies – and many of the smallest. He has a particular fascination for emerging technologies and is on a continuing mission to decompile, untangle, and explain techspeak to a wider audience, and excite people about the endless possibilities technology offers.

Devada, Inc.
600 Park Offices Drive
Suite 150
Research Triangle Park, NC

888.678.0399 919.678.0300