Monitor and Tune for Performance

SQL Server 2012 Books Online

Quick Reference

Microsoft

Monitor and Tune for Performance

SQL Server 2012 Books Online

Summary: The goal of monitoring databases is to assess how a server is performing. Effective monitoring involves taking periodic snapshots of current performance to isolate processes that are causing problems, and gathering data continuously over time to track performance trends. Ongoing evaluation of the database performance helps you minimize response times and maximize throughput, yielding optimal performance.

Category: Quick Reference
Applies to: SQL Server 2012
Source: SQL Server Books Online (link to source content)
E-book publication date: June 2012

Contents

Monitor and Tune for Performance

The goal of monitoring databases is to assess how a server is performing. Effective monitoring involves taking periodic snapshots of current performance to isolate processes that are causing problems, and gathering data continuously over time to track performance trends.

Ongoing evaluation of the database performance helps you minimize response times and maximize throughput, yielding optimal performance. Efficient network traffic, disk I/O, and CPU usage are key to peak performance. You need to thoroughly analyze the application requirements, understand the logical and physical structure of the data, assess database usage, and negotiate tradeoffs between conflicting uses such as online transaction processing (OLTP) versus decision support.

Benefits of Monitoring and Tuning Databases for Performance

Microsoft SQL Server and the Microsoft Windows operating system provide utilities that allow you to view the current condition of the database and to track performance as conditions change. There are a variety of tools and techniques that can be used to monitor Microsoft SQL Server. Understanding how to monitor SQL Server can help you:

- Determine whether you can improve performance. For example, by monitoring the response times for frequently used queries, you can determine whether changes to the query or indexes on the tables are required.
- Evaluate user activity. For example, by monitoring users trying to connect to an instance of SQL Server, you can determine whether security is set up adequately and test applications or development systems. For example, by monitoring SQL queries as they are executed, you can determine whether they are written correctly and producing the expected results.
- Troubleshoot any problems or debug application components, such as stored procedures.

Monitoring in a Dynamic Environment

Monitoring is important because SQL Server provides a service in a dynamic environment. Changing conditions result in changing performance. In your evaluations, you can see performance changes as the number of users increases, user access and connection methods change, database contents grow, client applications change, data in the applications changes, queries become more complex, and network traffic rises. By using SQL Server tools to monitor performance, you can associate some changes in performance with changing conditions and complex queries. The following scenarios provide examples:

- By monitoring the response times for frequently used queries, you can determine whether changes to the query or indexes on the tables where the queries execute are required.
- By monitoring Transact-SQL queries as they are executed, you can determine whether the queries are written correctly and producing the expected results.
- By monitoring users that try to connect to an instance of SQL Server, you can determine whether security is set up adequately and test applications or development systems.

Response time is the length of time required for the first row of the result set to be returned to the user in the form of visual confirmation that a query is being processed. Throughput is the total number of queries handled by the server during a specified period of time.

As the number of users increases, so does the competition for a server's resources, which in turn increases response time and decreases overall throughput.

Monitoring and Tuning Performance Tasks

Task Description	Topic
Provides the necessary steps required to effectively monitor any component of SQL Server.	Monitor SQL Server Components
Lists the SQL Server monitoring and tuning tools.	Performance Monitoring and Tuning Tools
Provides information about how to establish a performance baseline.	Establish a Performance Baseline
Describes how to isolate database performance problems.	Isolate Performance Problems
Describes how to monitor and track server performance to identify bottlenecks.	Identify Bottlenecks
Describes how to use SQL Server and Windows performance and activity monitoring tools.	Server Performance and Activity Monitoring
Describes how to display and save execution plans to a file in XML format.	Display and Save Execution Plans

See Also

Automating Administration Across an Enterprise

Database Engine Tuning Advisor

Monitoring Resource Usage (System Monitor)

SQL Server Profiler

Monitor SQL Server Components

Monitoring is important because SQL Server provides a service in a dynamic environment. The data in the application changes. The type of access that users require changes. The way that users connect changes. The types of applications accessing SQL Server may even change, but SQL Server automatically manages system-level resources, such as memory and disk space, to minimize the need for extensive system-level manual tuning. Monitoring lets administrators identify performance trends to determine if changes are necessary.

To monitor any component of SQL Server effectively:

1. Determine your monitoring goals.
2. Select the appropriate tool.
3. Identify components to monitor.
4. Select metrics for those components.
5. Monitor the server.
6. Analyze the data.

These steps are discussed in turn below.

Determine Your Monitoring Goals

To monitor SQL Server effectively you should clearly identify your reason for monitoring. Reasons can include the following:

- Establish a baseline for performance.
- Identify performance changes over time.
- Diagnose specific performance problems.
- Identify components or processes to optimize.
- Compare the effect of different client applications on performance.
- Audit user activity.
- Test a server under different loads.
- Test database architecture.
- Test maintenance schedules.

- Test backup and restore plans.
- Determining when to modify your hardware configuration.

Select the Appropriate Tool

After determining why you are monitoring, you should select the appropriate tools for that type of monitoring. The Windows operating system and SQL Server provide a complete set of tools to monitor servers in transaction-intensive environments. These tools clearly reveal the condition of an instance of the SQL Server Database Engine or an instance of SQL Server Analysis Services.

Windows provides the following tools for monitoring applications that are running on a server:

- System Monitor, which lets you collect and view real-time data about activities such as memory, disk, and processor usage
- Performance logs and alerts
- Task Manager

For more information about Windows Server or Windows tools, see the Windows documentation.

SQL Server provides the following tools for monitoring components of SQL Server:

- SQL Trace
- SQL Server Profiler
- Distributed Replay Utility
- SQL Server Management Studio Activity Monitor
- SQL Server Management Studio Graphical Showplan
- Stored procedures
- Database Console Commands (DBCC)
- Built-in functions
- Trace flags

For more information about SQL Server monitoring tools, see Tools for Monitoring Performance.

Identify the Components to Monitor

The third step to monitoring an instance of SQL Server is to identify the components that you monitor. For example, if you are using SQL Server Profiler to trace a server you can define the trace to collect data about specific events. You can also exclude events that do not apply to your situation.

Select Metrics for Monitored Components

After identifying the components to monitor, determine the metrics for components you monitor. For example, after selecting the events to include in a trace, you can choose to

include only specific data about the events. Limiting the trace to data that is relevant to the trace minimizes the system resources required to perform the tracing.

Monitor the Server

To monitor the server, run the monitoring tool that you have configured to gather data. For example, after a trace is defined, you can run the trace to gather data about events raised in the server.

Analyze the Data

After the trace has finished, analyze the data to see if you have achieved your monitoring goal. If you have not, modify the components or metrics that you used to monitor the server.

The following outlines the process for capturing event data and putting it to use.

1. Apply filters to limit the event data collected.

   Limiting the event data allows for the system to focus on the events pertinent to the monitoring scenario. For example, if you want to monitor slow queries, you can use a filter to monitor only those queries issued by the application that take more than 30 seconds to run against a particular database. For more information, see How to: Set a Trace Filter (Transact-SQL) and How to: Filter Events in a Trace (SQL Profiler).

2. Monitor (capture) events.

   As soon as it is enabled, active monitoring captures data from the specified application, instance of SQL Server, or operating system. For example, when disk activity is monitored using System Monitor, monitoring captures event data, such as disk reads and writes, and displays it on the screen. For more information, see Monitoring Resource Usage (System Monitor).

3. Save captured event data.

   Saving captured event data lets you analyze it later or even replay it using the Distributed Replay Utility or SQL Server Profiler. Captured event data is saved to a file that can be loaded back into the tool that originally created it for analysis. SQL Server Profiler permits event data to be saved to a SQL Server table. Saving captured event data is important when you are creating a performance baseline. The performance baseline data is saved and used, when comparing recently captured event data, to determine whether performance is optimal. For more information, see Using SQL Profiler.

4. Create trace templates that contain the settings specified to capture the events.

   Trace templates include specifications about the events themselves, event data, and filters that are used to capture data. These templates can be used to monitor a specific set of events later without redefining the events, event data, and filters. For example, if you want to frequently monitor the number of deadlocks, and the users involved in those deadlocks, you can create a template defining those events, event

data, and event filters; save the template; and reapply the filter the next time that you want to monitor deadlocks. SQL Server Profiler uses trace templates for this purpose. For more information, see How to: Set Trace Definition Defaults (SQL Profiler) and How to: Create a Trace Template (SQL Profiler).

5. Analyze captured event data.

   To be analyzed, the captured event data is loaded into the application that captured the data. For example, a captured trace from SQL Server Profiler can be reloaded into SQL Server Profiler for viewing and analysis. For more information, see Viewing and Analyzing Traces with SQL Profiler.

   Analyzing event data involves determining what is occurring and why. This information lets you make changes that can improve performance, such as adding more memory, changing indexes, correcting coding problems with Transact-SQL statements or stored procedures, and so on, depending on the type of analysis performed. For example, you can use the Database Engine Tuning Advisor to analyze a captured trace from SQL Server Profiler and make index recommendations based on the results.

6. Replay captured event data.

   Event replay lets you establish a test copy of the database environment from which the data was captured, and then repeat the captured events as they occurred originally on the real system. This capability is only available with the Distributed Replay Utility or SQL Server Profiler. You can replay the events at the same speed as they originally occurred, as fast as possible (to stress the system), or more likely, one step at a time (to analyze the system after each event has occurred). By analyzing the exact events in a test environment, you can prevent harm to the production system. For more information, see Replaying Traces.

Performance Monitoring and Tuning Tools

Microsoft SQL Server provides a comprehensive set of tools for monitoring events in SQL Server and for tuning the physical database design. The choice of tool depends on the type of monitoring or tuning to be done and the particular events to be monitored.

Following are the SQL Server monitoring and tuning tools:

Tool	Description
sp_trace_setfilter (Transact-SQL)	SQL Server Profiler tracks engine process events, such as the start of a batch or a transaction, enabling you to monitor server and database activity (for example, deadlocks, fatal errors, or login activity).

Tool	Description
	You can capture SQL Server Profiler data to a SQL Server table or a file for later analysis, and you can also replay the events captured on SQL Server step by step, to see exactly what happened.
Distributed Replay	Microsoft SQL Server Distributed Replay can use multiple computers to replay trace data, simulating a mission-critical workload.
System Monitor	System Monitor primarily tracks resource usage, such as the number of buffer manager page requests in use, enabling you to monitor server performance and activity using predefined objects and counters or user-defined counters to monitor events. System Monitor (Performance Monitor in Microsoft Windows NT 4.0) collects counts and rates rather than data about the events (for example, memory usage, number of active transactions, number of blocked locks, or CPU activity). You can set thresholds on specific counters to generate alerts that notify operators. System Monitor works on Microsoft Windows Server and Windows operating systems. It can monitor (remotely or locally) an instance of SQL Server on Windows NT 4.0 or later. The key difference between SQL Server Profiler and System Monitor is that SQL Server Profiler monitors Database Engine events, whereas System Monitor monitors resource usage associated with server processes.
Activity Monitor	The Activity Monitor in SQL Server Management Studio graphically displays information about: • Processes running on an instance of SQL Server.

<table>
<tr><th>Tool</th><th>Description</th></tr>
<tr><td></td><td>• Blocked processes.
• Locks.
• User activity.
This is useful for ad hoc views of current activity.</td></tr>
<tr><td>SQL Trace</td><td>Transact-SQL stored procedures that create, filter, and define tracing:
• sp_trace_create
• sp_trace_generateevent
• sp_trace_setevent
• sp_trace_setfilter
• sp_trace_setstatus</td></tr>
<tr><td>Error Logs</td><td>The Windows application event log provides an overall picture of events occurring on the Windows Server and Windows operating systems as a whole, as well as events in SQL Server, SQL Server Agent, and full-text search. It contains information about events in SQL Server that is not available elsewhere. You can use the information in the error log to troubleshoot SQL Server-related problems.</td></tr>
<tr><td>System Stored Procedures</td><td>The following SQL Server system stored procedures provide a powerful alternative for many monitoring tasks:
<table><tr><th>Stored procedure</th><th>Description</th></tr><tr><td>sp_who</td><td>Reports snapshot information about current SQL Server users and processes, including the currently executing statement and whether the statement is</td></tr></table></td></tr>
</table>

<table>
<tr><th>Tool</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>blocked.</td></tr>
<tr><td>sp_lock</td><td>Reports snapshot information about locks, including the object ID, index ID, type of lock, and type or resource to which the lock applies.</td></tr>
<tr><td>sp_spaceused</td><td>Displays an estimate of the current amount of disk space used by a table (or a whole database).</td></tr>
<tr><td>sp_monitor</td><td>Displays statistics, including CPU usage, I/O usage, and the amount of time idle since sp_monitor was last executed.</td></tr>
</table>
</td></tr>
<tr><td>DBCC statements</td><td>DBCC (Database Console Command) statements enable you to check performance statistics and the logical and physical consistency of a database.</td></tr>
<tr><td>Built-in Functions</td><td>Built-in functions display snapshot statistics about SQL Server activity since the server was started; these statistics are stored in predefined SQL Server counters. For example, @@CPU_BUSY contains the amount of time the CPU has been executing SQL Server code; @@CONNECTIONS contains the number of SQL Server connections or attempted</td></tr>
</table>

Tool	Description
	connections; and **@@PACKET_ERRORS** contains the number of network packets occurring on SQL Server connections.
Trace Flags	Trace flags display information about a specific activity within the server and are used to diagnose problems or performance issues (for example, deadlock chains).
Database Engine Tuning Advisor	Database Engine Tuning Advisor analyzes the performance effects of Transact-SQL statements executed against databases you want to tune. Database Engine Tuning Advisor provides recommendations to add, remove, or modify indexes, indexed views, and partitioning.

Choosing a Monitoring Tool

The choice of a monitoring tool depends on the event or activity to be monitored.

Event or activity	SQL Server Profiler	Distributed Replay	System Monitor	Activity Monitor	Transact-SQL	Error logs
Trend analysis	Yes		Yes			
Replaying captured events	Yes (From a single computer)	Yes (From multiple computers)				
Ad hoc monitoring	Yes			Yes	Yes	Yes
Generating alerts			Yes			
Graphical interface	Yes		Yes	Yes		Yes
Using within custom application	Yes [1]				Yes	

[1] Using SQL Server Profiler system stored procedures.

Windows Monitoring Tools

Windows operating systems and Windows Server 2003 also provide these monitoring tools.

Tool	Description
Task Manager	Shows a synopsis of the processes and applications running on the system.
Network Monitor Agent	Monitors network traffic.

For more information about Windows operating systems or Windows Server tools, see the Windows documentation.

Establish a Performance Baseline

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.

See Also

sp_configure (Transact-SQL)

Isolate Performance Problems

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.

Tuning the Physical Database Design

Monitoring Memory Usage

Identify Bottlenecks

Simultaneous access to shared resources causes bottlenecks. In general, bottlenecks are present in every software system and are inevitable. However, excessive demands on shared resources cause poor response time and must be identified and tuned.

Causes of bottlenecks include:

- Insufficient resources, requiring additional or upgraded components.
- Resources of the same type among which workloads are not distributed evenly; for example, one disk is being monopolized.
- Malfunctioning resources.
- Incorrectly configured resources.

Analyzing Bottlenecks

Excessive durations for various events are indicators of bottlenecks that can be tuned.

For example:

- Some other component may prevent the load from reaching this component thereby increasing the time to complete the load.
- Client requests may take longer due to network congestion.

Following are five key areas to monitor when tracking server performance to identify bottlenecks.

Possible bottleneck area	Effects on the server
Memory usage	Insufficient memory allocated or available to Microsoft SQL Server degrades performance. Data must be read from the disk rather than directly from the data cache. Microsoft Windows operating systems perform excessive paging by swapping data to and from the disk as the pages are needed.
CPU utilization	A chronically high CPU utilization rate may indicate that Transact-SQL queries need to be tuned or that a CPU upgrade is needed.
Disk input/output (I/O)	Transact-SQL queries can be tuned to reduce unnecessary I/O; for example, by employing indexes.
User connections	Too many users may be accessing the

Possible bottleneck area	Effects on the server
	server simultaneously causing performance degradation.
Blocking locks	Incorrectly designed applications can cause locks and hamper concurrency, thus causing longer response times and lower transaction throughput rates.

See Also

Monitoring CPU Use

Monitoring Disk Activity

Monitoring Memory Usage

SQL Server: General Statistics Object

SQL Server: Locks Object

Server Performance and Activity Monitoring

The goal of monitoring databases is to assess how a server is performing. Effective monitoring involves taking periodic snapshots of current performance to isolate processes that are causing problems, and gathering data continuously over time to track performance trends. Microsoft SQL Server and the Microsoft Windows operating system provide utilities that let you view the current condition of the database and to track performance as conditions change.

The following section contains topics that describe how to use SQL Server and Windows performance and activity monitoring tools. It contains the following topics:

In This Section

To perform monitoring tasks with Windows tools

- How to: Start System Monitor (Windows)
- How to: View the Windows Application Log (Windows)

To create SQL Server database alerts with Windows tools

- How to: Set up a SQL Server Database Alert (Windows)

To perform monitoring tasks with SQL Server Management Studio

- How to: View the SQL Server Error Log (SQL Server Management Studio)
- How to: Open Activity Monitor (SQL Server Management Studio)

To perform monitoring tasks with SQL Trace by using Transact-SQL stored procedures

- How to: Create a Trace (Transact-SQL)
- How to: Set a Trace Filter (Transact-SQL)
- How to: Modify an Existing Trace (Transact-SQL)
- How to: View a Saved Trace (Transact-SQL)
- How to: View Filter Information (Transact-SQL)
- How to: Delete a Trace (Transact-SQL)

To create and modify traces by using SQL Server Profiler

- How to: Create a Trace (SQL Profiler)
- How to: Set Global Trace Options (SQL Server Profiler)
- How to: Specify Events and Data Columns for a Trace File (SQL Profiler)
- How to: Create a SQL Script for Running a Trace (SQL Server Profiler)
- How to: Save Trace Results to a File (SQL Profiler)
- How to: Set a Maximum File Size for a Trace File (SQL Profiler)
- How to: Save Trace Results to a Table (SQL Profiler)
- How to: Set a Maximum Table Size for a Trace Table (SQL Profiler)
- How to: Filter Events in a Trace (SQL Profiler)
- How to: View Filter Information (SQL Profiler)
- How to: Modify a Filter (SQL Profiler)
- How to: Filter Events Based on the Event Start Time (SQL Profiler)
- How to: Filter Events Based on the Event End Time (SQL Profiler)
- How to: Filter System IDs in a Trace (SQL Profiler)
- How to: Organize Columns Displayed in a Trace (SQL Server Profiler)

To start, pause, and stop traces by using SQL Server Profiler

- How to: Start a Trace Automatically after Connecting to a Server (SQL Profiler)
- How to: Pause a Trace (SQL Profiler)
- How to: Stop a Trace (SQL Profiler)
- How to: Run a Trace After It Has Been Paused or Stopped (SQL Profiler)

To open traces and configure how traces are displayed by using SQL Server Profiler

- How to: Open a Trace File (SQL Profiler)
- How to: Open a Trace Table (SQL Profiler)
- How to: Clear a Trace Window (SQL Profiler)
- How to: Close a Trace Window (SQL Profiler)
- How to: Set Trace Definition Defaults (SQL Profiler)
- How to: Set Trace Display Defaults (SQL Profiler)

To replay traces by using SQL Server Profiler

- How to: Replay a Trace File (SQL Profiler)
- How to: Replay a Trace Table (SQL Profiler)
- How to: Replay a Single Event at a Time (SQL Server Profiler)
- How to: Replay to a Breakpoint (SQL Server Profiler)
- How to: Replay to a Cursor (SQL Server Profiler)
- How to: Replay an SQL Script (SQL Server Profiler)

To create, modify, and use trace templates by using SQL Server Profiler

- How to: Create a Trace Template (SQL Server Profiler)
- How to: Modify a Trace Template (SQL Server Profiler)
- How to: Derive a Template from a Running Trace (SQL Server Profiler)
- How to: Derive a Template from a Trace File or Trace Table (SQL Server Profiler)
- How to: Export a Trace Template (SQL Server Profiler)
- How to: Import a Trace Template (SQL Server Profiler)

To use SQL Server Profiler traces to collect and monitor server performance

- How to: Find a Value or Data Column While Tracing (SQL Server Profiler)
- How to: Save Deadlock Graphs (SQL Server Profiler)
- How to: Save Showplan XML Events Separately (SQL Server Profiler)
- How to: Save Showplan XML Statistics Profile Events Separately (SQL Server Profiler)
- How to: Extract a Script from a Trace (SQL Server Profiler)
- How to: Correlate a Trace with Windows Performance Log Data (SQL Server Profiler)

Start System Monitor (Windows)

Use System Monitor to monitor the utilization of system resources. Collect and view real-time performance data in the form of counters, for server resources such as processor and memory use, and for many Microsoft SQL Server resources such as locks and transactions.

Procedures

To start System Monitor in Windows

1. On the **Start** menu, point to **Run**, type **perfmon** in the Run dialog box, and then click **OK**.

See Also

Running System Monitor

Set Up a SQL Server Database Alert (Windows)

Using System Monitor, you can create an alert to be raised when a threshold value for a System Monitor counter has been reached. In response to the alert, System Monitor can launch an application, such as a custom application written to handle the alert condition. For example, you can create an alert that is raised when the number of deadlocks exceeds a specific value.

Alerts also can be defined using Microsoft SQL Server Management Studio and SQL Server Agent. For more information, see Creating a SQL Server Database Alert.

Procedures

To set up a SQL Server database alert

1. On the navigation tree of the Performance window, expand **Performance Logs and Alerts**.
2. Right-click **Alerts**, and then click **New Alert Settings**.
3. In the **New Alert Settings** dialog box, type a name for the new alert, and then click **OK**.
4. On the **General** tab of the dialog box for the new alert, add a **Comment**, and click **Add** to add a counter to the alert.

   All alerts must have at least one counter.
5. In the Add Counters dialog box, select a SQL Server object from the **Performance Object** list, and then select a counter from the **Select counters from list**.
6. To add the counter to the alert, click **Add**. You can continue to add counters, or you can click **Close** to return to the dialog box for the new alert.
7. In the new alert dialog box, click either **Over** or **Under** in the **Alert when the value is** list, and then enter a threshold value in **Limit**.

   The alert is generated when the value for the counter is more than or less than the threshold value (depending on whether you clicked **Over** or **Under**).
8. In the **Sample data every** boxes, set the sampling frequency.
9. On the **Action** tab, set actions to occur every time the alert is triggered.
10. On the **Schedule** tab, set the start and stop schedule for the alert scan.

See Also

Creating a SQL Server Database Alert

View the Windows Application Log (Windows)

When SQL Server is configured to use the Windows application log, each SQL Server session writes new events to that log. Unlike the SQL Server error log, a new application log is not created each time you start an instance of SQL Server.

Procedures

To view the Windows application log

1. On the **Start** menu, point to **All Programs**, point to **Administrative Tools**, and then click **Event Viewer**.
2. In Event Viewer, click **Application**.
3. SQL Server events are identified by the entry **MSSQLSERVER** (named instances are identified with **MSSQL$**<*instance_name*>) in the **Source** column. SQL Server Agent events are identified by the entry SQLSERVERAGENT (for named instances of SQL Server, SQL Server Agent events are identified with **SQLAgent$**<*instance_name*>). Microsoft Search service events are identified by the entry **Microsoft Search**.
4. To view the log of a different computer, right-click **Event Viewer**, click **Connect to another computer,** and complete the **Select Computer** dialog box.
5. Optionally, to display only SQL Server events, on the **View** menu click **Filter**, and in the **Event source** list, select **MSSQLSERVER**. To view only SQL Server Agent events, instead select **SQLSERVERAGENT** in the **Event source** list.
6. To view more information about an event, double-click the event.

See Also

Viewing the Windows Application Log

View the SQL Server Error Log (SQL Server Management Studio)

The SQL Server error log contains user-defined events and certain system events. You can use this error log to troubleshoot problems related to SQL Server.

Procedures

To view the SQL Server error log

1. In **Object Explorer**, expand a server, expand **Management**, and then expand **SQL Server Logs**.
2. Right-click a log and click **View SQL Server Log**.

Save Deadlock Graphs (SQL Server Profiler)

This topic describes how to save a deadlock graph by using SQL Server Profiler. Deadlock graphs are saved as XML files.

Procedures

To save deadlock graph events separately

1. On the **File** menu, click **New Trace**, and then connect to an instance of SQL Server.

   The **Trace Properties** dialog box appears.

   > **Note**
   > If **Start tracing immediately after making connection** is selected, the **Trace Properties** dialog box fails to appear, and the trace begins instead. To turn off this setting, on the **Tools** menu, click **Options**, and clear the **Start tracing immediately after making connection** check box.

2. In the Trace Properties dialog box, type a name for the trace in the **Trace name** box.
3. In the **Use the template** list, select a trace template on which to base the trace, or select **Blank** if you do not want to use a template.
4. Do one of the following:
   - Select the **Save to file** check box to capture the trace to a file. Specify a value for **Set maximum file size**.

     Optionally, select **Enable file rollover** and **Server processes trace data**.
   - Select the **Save to table** check box to capture the trace to a database table.

     Optionally, click **Set maximum rows**, and specify a value.
5. Optionally, select the **Enable trace stop time** check box, and specify a stop date and time.
6. Click the **Events Selection** tab.
7. In the **Events** data column, expand the **Locks** event category, and then select the **Deadlock graph** check box. If the Locks event category is not available, check **Show all events** to display it.

   The **Events Extraction Settings** tab is added to the **Trace Properties** dialog box.
8. On the **Events Extraction Settings** tab, click **Save Deadlock XML Events Separately**.
9. In the **Save As** dialog box, enter the name of the file in which to store the deadlock graph events.

10. Click **All Deadlock XML batches in a single file** to save all deadlock graph events in a single XML file, or click **Each Deadlock XML batch in a distinct file** to create a new XML file for each deadlock graph.

After you have saved the deadlock file, you can open the file in SQL Server Management Studio. For more information, see Analyzing Deadlocks with SQL Server Profiler.

See Also

Analyzing Deadlocks with SQL Server Profiler

Open, View, and Print a Deadlock File (SQL Server Management Studio)

When SQL Server Profiler generates a deadlock, you can capture and save the deadlock information to a file. After you have saved the deadlock file, you can open it in SQL Server Management Studio to view or print.

Procedures

To open and view a deadlock file

1. On the **File** menu in SQL Server Management Studio, point to **Open**, and then click **File**.
2. In the **Open File** dialog box, select the .xdl file type in the **Files of type** box. You will now have a filtered list of only deadlock files.

To print a deadlock file

1. On the **File** menu in SQL Server Management Studio, point to **Open,** and then click **File**.
2. In the **Open File** dialog box, select the .xdl file type in the **Files of type** box. You will now have a filtered list of only deadlock files.
3. Select the deadlock file you want to print, and click **Open**.
4. On the **File** menu, click **Print.**

See Also

How to: Save Deadlock Graphs (SQL Server Profiler)

Save Showplan XML Events Separately (SQL Server Profiler)

This topic describes how to save **Showplan XML** events that are captured in traces into separate .SQLPlan files by using SQL Server Profiler. You can open the **Showplan XML** event files in SQL Server Management Studio, which enables you to view the graphical execution plan for each event.

Procedures

To save Showplan XML events separately

1. On the **File** menu, click **New Trace**, and then connect to an instance of SQL Server.

   The **Trace Properties** dialog box appears.

   Note
   If **Start tracing immediately after making connection** is selected, the **Trace Properties** dialog box fails to appear and the trace begins instead. To turn off this setting, on the **Tools** menu, click **Options**, and clear the **Start tracing immediately after making connection** check box.
2. In the **Trace Properties** dialog box, type a name for the trace in the **Trace name** box.
3. In the **Use the template** list, select a trace template on which to base the trace, or select **Blank** if you do not want to use a template.
4. Do one of the following:
   - Select the **Save to file** check box to capture the trace to a file. Specify a value for **Set maximum file size**. Optionally, select the **Enable file rollover** and **Server processes trace data** check boxes.
   - Select the **Save to table** check box to capture the trace to a database table. Optionally click **Set maximum rows**, and specify a value.
5. Optionally, select the **Enable trace stop time** check box, and specify a stop date and time.
6. Click the **Events Selection** tab.
7. In the **Events** data column, expand the **Performance** event category, and then select the **Showplan XML** check box. If the **Performance** event category is not available, check **Show all events** to display it.

   The **Events Extraction Settings** tab is added to the **Trace Properties** dialog box.
8. On the **Events Extraction Settings** tab, click **Save XML Showplan events separately**.
9. In the **Save As** dialog box, enter the name of the file in which to store the

Showplan XML events.

10. Click **All XML Showplan batches in a single file** to save all **Showplan XML** events in a single XML file, or click **Each XML Showplan batch in a distinct file** to create a new XML file for each **Showplan XML** event.
11. To view the **Showplan XML** event file in SQL Server Management Studio, on the **File** menu, point to **Open**, and click **File**. Navigate to the directory where you saved the **Showplan XML** event file or files to select one and open it. **Showplan XML** event files have a .SQLPlan file extension.

See Also

Analyzing Queries with SHOWPLAN Results in SQL Server Profiler

Save Showplan XML Statistics Profile Events Separately (SQL Server Profiler)

This topic describes how to save **Showplan XML Statistics Profile** events that are captured in traces into separate .SQLPlan files by using SQL Server Profiler. You can open the **Showplan XML Statistics Profile** event files in SQL Server Management Studio, which enables you to view the graphical execution plan for each event.

Procedures

To save Showplan XML statistics events separately

1. On the **File** menu, click **New Trace**, and then connect to an instance of SQL Server.

   The **Trace Properties** dialog box appears.

   Note
   If **Start tracing immediately after making connection** is selected, the **Trace Properties** dialog box does not appear and the trace begins instead. To turn off this setting, on the **Tools** menu, click **Options**, and clear the **Start tracing immediately after making connection** check box.
2. In the **Trace Properties** dialog box, type a name for the trace in the **Trace name** box.
3. In the **Use the template** list, select a trace template to base the trace on, or select **Blank** if you do not want to use a template.
4. Do one of the following:
   - Click **Save to file** to capture the trace to a file. Specify a value for **Set maximum file size**.

Optionally select **Enable file rollover** and **Server processes trace data**.

- Click **Save to table** to capture the trace to a database table.

  Optionally click **Set maximum rows**, and specify a value.

5. Optionally select the **Enable trace stop time** check box, and specify a stop date and time.
6. Click the **Events Selection** tab.
7. In the **Events** data column, expand the **Performance** event category, and then select the **Showplan XML Statistics Profile** check box. If the **Performance** event category is not available, check **Show all events** to display it.

   The **Events Extraction Settings** tab is added to the **Trace Properties** dialog.
8. On the **Events Extraction Settings** tab, click **Save XML Showplan events separately**.
9. In the **Save As** dialog box, enter the file name to store the **Showplan XML Statistics Profile** events.
10. Click **All batches in a single file** to save all **Showplan XML Statistics Profile** events in a single XML file, or click **Each XML Showplan batch in a distinct file** to create a new XML file for each **Showplan XML Statistics Profile** event.
11. To view the **Showplan XML Statistics Profile** event file in SQL Server Management Studio, on the **File** menu, point to **Open**, and click **File**. Navigate to the directory where you saved the **Showplan XML Statistics Profile** event file or files to select one and open it. **Showplan XML Statistics Profile** event files have a .SQLPlan file extension.

See Also

Analyzing Queries with SHOWPLAN Results in SQL Server Profiler

Display and Save Execution Plans

This section explains how to display execution plans and how to save execution plans to a file in XML format by using Microsoft SQL Server Management Studio.

Execution plans graphically display the data retrieval methods chosen by the SQL Server query optimizer. Execution plans represent the execution cost of specific statements and queries in SQL Server using icons rather than the tabular representation produced by the SET SHOWPLAN_ALL or SET SHOWPLAN_TEXT statements. This graphical approach is very useful for understanding the performance characteristics of a query.

In this Section

- How to: Display the Estimated Execution Plan
- How to: Display an Actual Execution Plan

- How to: Save an Execution Plan in XML Format

Display the Estimated Execution Plan

This topic describes how to generate graphical estimated execution plans by using SQL Server Management Studio. When estimated execution plans are generated, the Transact-SQL queries or batches do not execute. Instead, the execution plan that is generated displays the query execution plan that SQL Server Database Engine would most probably use if the queries were actually executed.

To use this feature, users must have the appropriate permissions to execute the Transact-SQL query for which a graphical execution plan is being generated, and they must be granted the SHOWPLAN permission for all databases referenced by the query.

Procedures

To display the estimated execution plan for a query

1. On the toolbar, click **Database Engine Query**. You can also open an existing query and display the estimated execution plan by clicking the **Open File** toolbar button and locating the existing query.
2. Enter the query for which you would like to display the estimated execution plan.
3. On the **Query** menu, click **Display Estimated Execution Plan** or click the **Display Estimated Execution Plan** toolbar button. The estimated execution plan is displayed on the **Execution Plan** tab in the results pane. To view additional information, pause the mouse over the logical and physical operator icons and view the description and properties of the operator in the displayed ToolTip. Alternatively, you can view operator properties in the Properties window. If Properties is not visible, right-click an operator and click **Properties**. Select an operator to view its properties.
4. To alter the display of the execution plan, right-click the execution plan and select **Zoom In**, **Zoom Out**, **Custom Zoom**, or **Zoom to Fit**. **Zoom In** and **Zoom Out** allow you to magnify or reduce the execution plan by fixed amounts. **Custom Zoom** allows you to define your own display magnification, such as zooming at 80 percent. **Zoom to Fit** magnifies the execution plan to fit the result pane.

Display an Actual Execution Plan

This topic describes how to generate actual graphical execution plans by using SQL Server Management Studio. When actual execution plans are generated, the Transact-

SQL queries or batches execute. The execution plan that is generated displays the actual query execution plan that the SQL Server Database Engine uses to execute the queries.

To use this feature, users must have the appropriate permissions to execute the Transact-SQL queries for which a graphical execution plan is being generated, and they must be granted the SHOWPLAN permission for all databases referenced by the query.

Procedures

To include an execution plan for a query during execution

1. On the SQL Server Management Studio toolbar, click **Database Engine Query**. You can also open an existing query and display the estimated execution plan by clicking the **Open File** toolbar button and locating the existing query.
2. Enter the query for which you would like to display the actual execution plan.
3. On the **Query** menu, click **Include Actual Execution Plan** or click the **Include Actual Execution Plan** toolbar button
4. Execute the query by clicking the **Execute** toolbar button. The plan used by the query optimizer is displayed on the **Execution Plan** tab in the results pane. Pause the mouse over the logical and physical operators to view the description and properties of the operators in the displayed ToolTip.

   Alternatively, you can view operator properties in the Properties window. If Properties is not visible, right-click an operator and select **Properties**. Select an operator to view its properties.
5. You can alter the display of the execution plan by right-clicking the execution plan and selecting **Zoom In**, **Zoom Out**, **Custom Zoom**, or **Zoom to Fit**. **Zoom In** and **Zoom Out** allow you to zoom in or out on the execution plan, while **Custom Zoom** allows you to define your own zoom, such as zooming at 80 percent. **Zoom to Fit** magnifies the execution plan to fit the result pane.

Save an Execution Plan in XML Format

Use SQL Server Management Studio to save execution plans as an XML file, and to open them for viewing.

To use the execution plan feature in Management Studio, or to use the XML Showplan SET options, users must have the appropriate permissions to execute the Transact-SQL query for which an execution plan is being generated, and they must be granted the SHOWPLAN permission for all databases referenced by the query.

Procedures

To save a query plan by using the XML Showplan SET options

1. In SQL Server Management Studio open a query editor and connect to Database Engine.
2. Turn SHOWPLAN_XML on with the following statement:

   ```
   SET SHOWPLAN_XML ON;
   GO
   ```

   To turn STATISTICS XML on, use the following statement:

   ```
   SET STATISTICS XML ON;
   GO
   ```

   SHOWPLAN_XML generates compile-time query execution plan information for a query, but does not execute the query. STATISTICS XML generates run-time query execution plan information for a query, and executes the query.
3. Execute a query. Example:

   ```
   USE AdventureWorks2012;
   GO
   SET SHOWPLAN_XML ON;
   GO
   -- Execute a query.
   SELECT BusinessEntityID
   FROM HumanResources.Employee
   WHERE NationalIDNumber = '509647174';
   GO
   SET SHOWPLAN_XML OFF;
   ```
4. In the **Results** pane, right-click the **Microsoft SQL Server XML Showplan** that contains the query plan, and then click **Save Results As**.
5. In the **Save** <Grid or Text> **Results** dialog box, in the **Save as type** box, click **All files (*.*)**.
6. In the **File name** box provide a name, in the format <name**>.sqlplan**, and then click **Save**.

To save an execution plan by using SQL Server Management Studio options

1. Generate either an estimated execution plan or an actual execution plan by using Management Studio. For more information, see How to: Display the Estimated Execution Plan or How to: Display an Actual Execution Plan.
2. In the **Execution plan** tab of the results pane, right-click the graphical execution plan, and choose **Save Execution Plan As**.

   As an alternative, you can also choose **Save Execution Plan As** on the **File** menu.

3. In the **Save As** dialog box, make sure that the **Save as type** is set to **Execution Plan Files (*.sqlplan)**.
4. In the **File name** box provide a name, in the format <name>**.sqlplan**, and then click **Save**.

To open a saved XML query plan in SQL Server Management Studio

1. In SQL Server Management Studio, on the **File** menu, choose **Open**, and then click **File**.
2. In the **Open File** dialog box, set **Files of type** to **Execution Plan Files (*.sqlplan)** to produce a filtered list of saved XML query plan files.
3. Select the XML query plan file that you want to view, and click **Open**.

   As an alternative, in Windows Explorer, double-click a file with extension **.sqlplan**. The plan opens in Management Studio.

See Also

SET SHOWPLAN_XML (Transact-SQL)

SET STATISTICS XML (Transact-SQL)