Monitoring Microsoft SQL

eG Enterprise v6.1.2

Table of contents

Table of Figures

1

Configuring Microsoft SQL server

1.1 User Privileges Required for Monitoring Microsoft SQL server

- To monitor an Microsoft SQL server 7.0/2000, all tests should be configured with the credentials of a SQL user with **Sysadmin** role.
- To execute the Microsoft SQL Database Space test of an Microsoft SQL server 2005/2008/2012, provide the credentials of a SQL user with **Sysadmin** role. All other tests should be configured with the credentials of a SQL user with **connect sql, view any database**, and **view server state** roles.

1.2 Configuring the Microsoft SQL Server to Communicate with the eG Manager via HTTP/HTTPS

1. The eG agent on the Microsoft SQL server communicates with the eG manager via HTTP/HTTPS. However, some Microsoft SQL server installations could, by default, support only TCP/IP connections. Under such circumstances, you might have to reconfigure the SQL installation to additionally support HTTP/HTTPS protocols, so that the eG agent-manager communication is not affected. To ensure this, do the following:
2. On the Microsoft SQL server host, follow the menu sequence, Start -> Programs -> Microsoft SQL Server -> Client Network Utility, to open the Microsoft SQL server's **Client Network Utility** (see Figure 1.1).

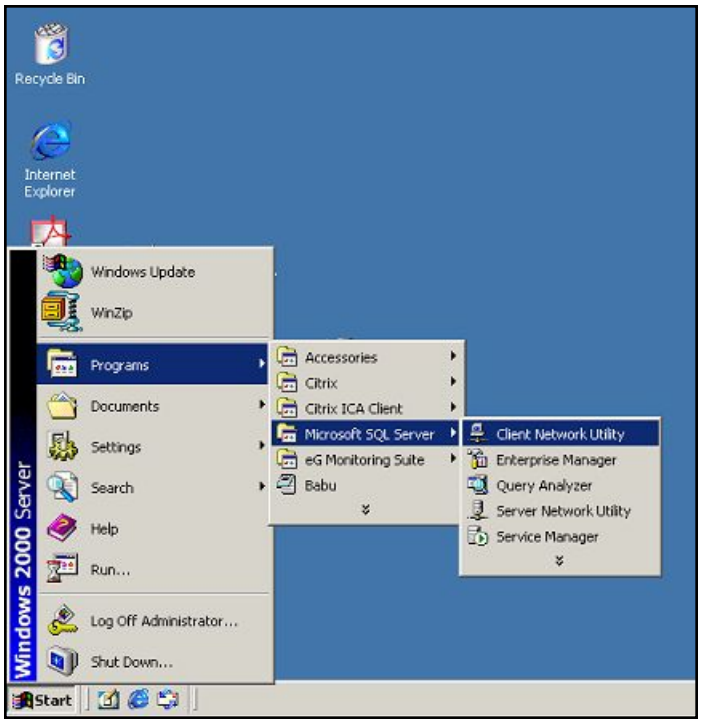


Figure 1.1: Opening the Client Network Utility

3. In the **General** tab of Figure 1.2 that appears next, check whether the **Multiprotocol** option is available in the **Enable protocols by order** list. If not, then select it from the **Disabled protocols** list and select the **Enable** button to enable it.

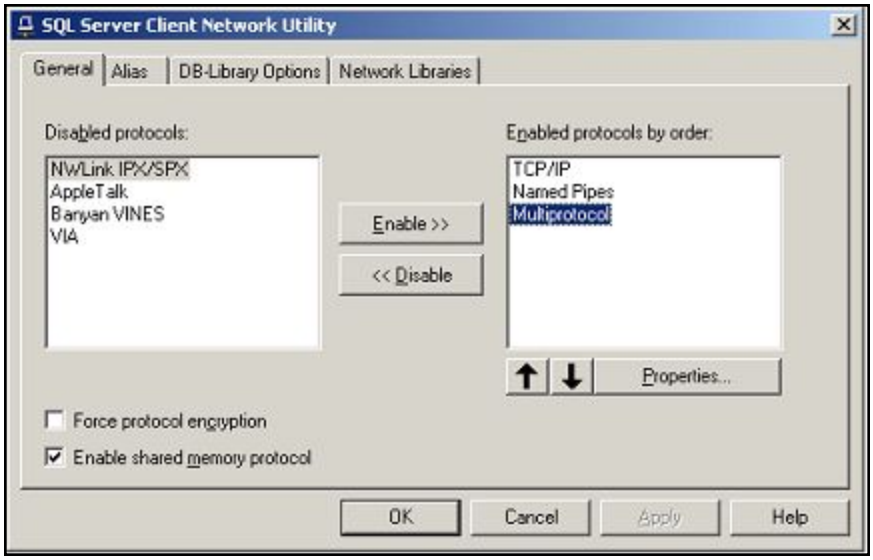


Figure 1.2: Enabling Multiprotocol support using the SQL Client Network Utility

4. Finally, click the **Apply** and **OK** buttons in Figure 1.2 to register the changes.

5. Next, follow the menu sequence depicted by Figure 1.3 to open the Microsoft SQL server's **Server Network Utility**.

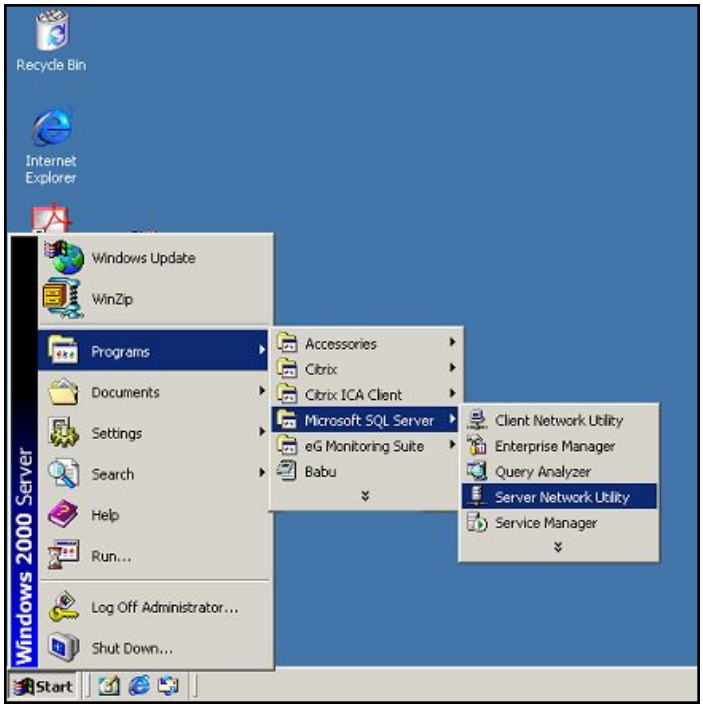


Figure 1.3: Opening the Server Network Utility

6. When Figure 1.4 appears, check whether the **Multiprotocol** option is available in the **Enable protocols by order** list. If not, then select it from the **Disabled protocols** list and select the **Enable** button to enable it.

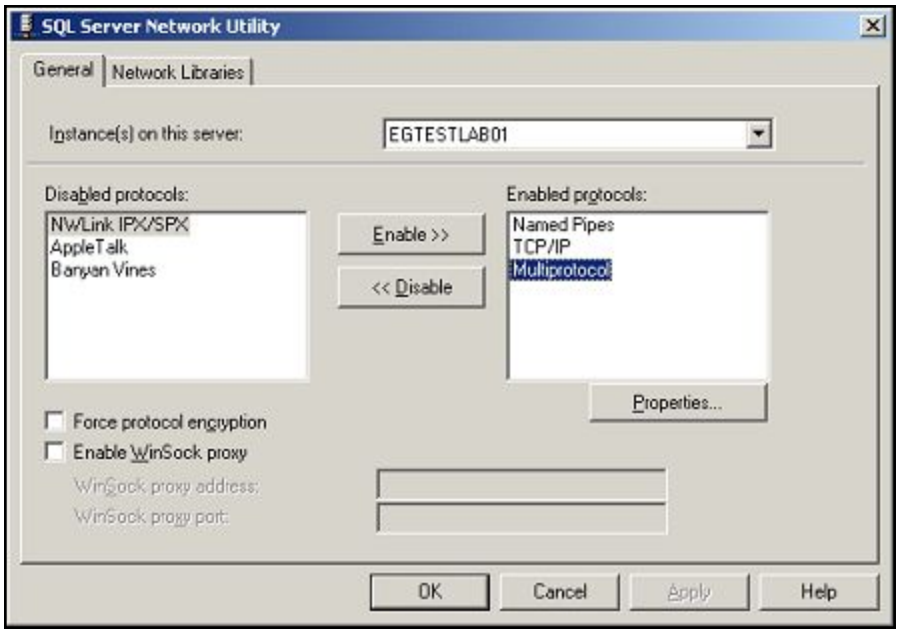


Figure 1.4: Enabling Multiprotocol support using the SQL Server Network Utility

7. Finally, click the **Apply** and **OK** buttons in Figure 1.4 to register the changes.

Note:

Before attempting to monitor an **instance** based server, make sure that the **SQL Browser Service** is up and running on the Microsoft SQL server.

2

Administering the eG Manager to monitor a Microsoft SQL Server

To configure eG to monitor a *Microsoft SQL* server, do the following:

1. First, login through the eG user interface.
2. If the Microsoft SQL Server is already discovered, use the menu sequence Infrastructure -> Components -> Manage/Unmanage to manage it (see Figure 2.1 and Figure 2.2 for a pictorial representation of the server management procedure). Otherwise, manually add the target Microsoft SQL server using the **COMPONENTS** page. The components so added are automatically managed.

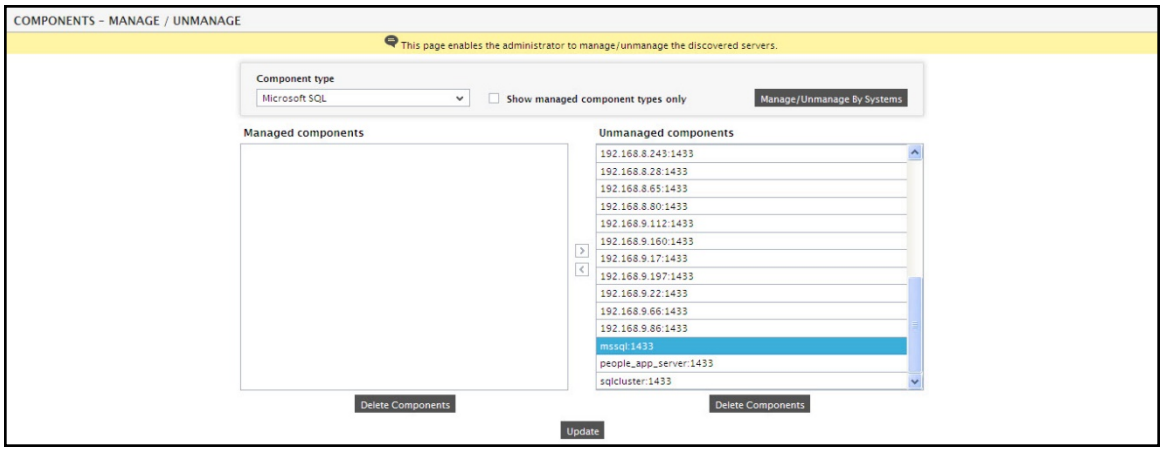


Figure 2.1: Selecting the component-type to be managed

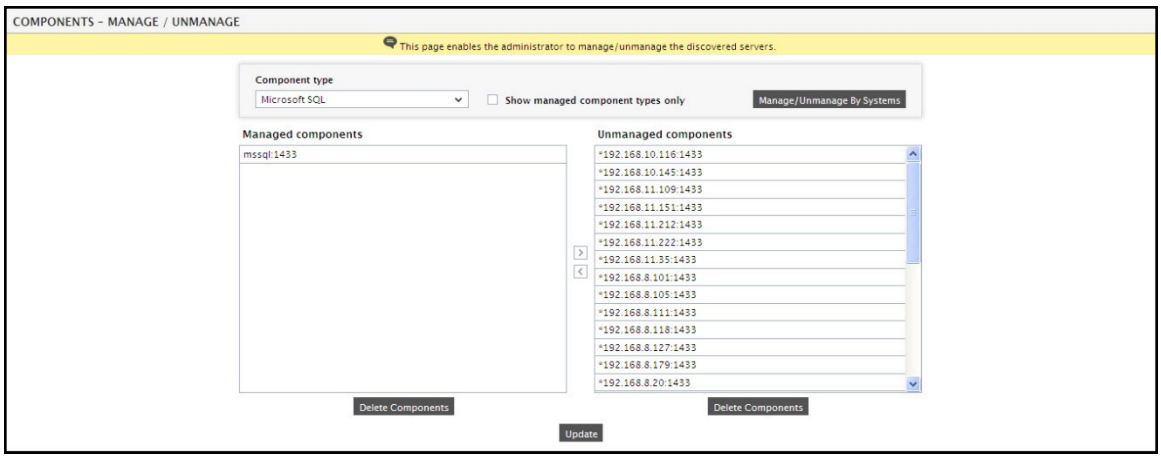


Figure 2.2: Managing / Unmanaging a Microsoft SQL Server

3. Next, sign out of the eG administrative interface. Along with other tests, you will be prompted to configure an SQL Network test (see Figure 2.3). This test emulates a user request to a Microsoft SQL Server and hence, requires the user name and password that it should use to be configured.

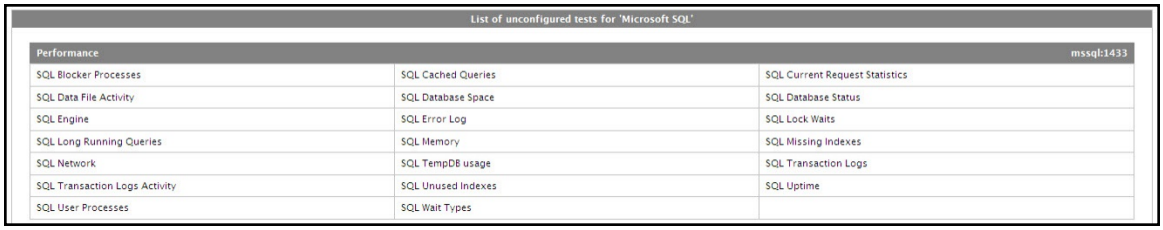


Figure 2.3: List of Unconfigured tests displaying the SQL Network test as unconfigured

4. Click on the **SQL Network** test to configure it. Doing so, will reveal test parameters. To know how to configure the test Click here.

3

Monitoring MS SQL Servers

Microsoft's SQL server has emerged as the database engine of choice for most applications hosted on the Microsoft Windows platform. Services in various domains - healthcare, manufacturing, banking, etc. - rely on the backend database servers for data storage and access. Any performance degradation or unavailability of the database servers can severely impact the performance of the entire service, often causing customer dissatisfaction and lost business revenue. Continuous monitoring of the MS SQL servers are hence imperative.

The pre-built MS SQL monitoring model that eG Enterprise offers (see Figure 3.1), provides in-depth monitoring for SQL database servers.

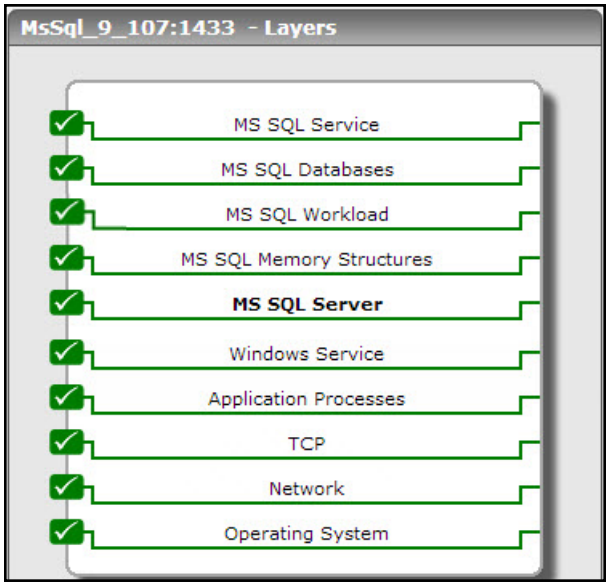


Figure 3.1: Layer model for MS SQL servers

Each of the layers of this hierarchical model reports a wide variety of metrics ranging from the basic operating system-level statistics to individual database related measurements to those indicating the database engine health. The table below sheds light on what the eG SQL Monitor reveals:

Database Service Monitoring	• Is the database server available for servicing requests? • What is the response time for a typical query? • How many logins/logouts are happening on the SQL server? • Which applications/users are accessing the SQL server and what is their respective resource usage? • What queries are each of the applications currently executing?

Database Server Engine Monitoring	• What is the CPU utilization of the database server engine? • How much time is the SQL server spending on processing vs. I/O? • What is the typical workload on the database server? • Which databases are imposing most load on the database server engine? • How many processes are running, and what queries are they executing? • Which user(s) are executing these queries?
Lock Activity Monitoring	• What is the typical locking activity on the database? • Which processes are being blocked and by whom? • Which are the root-blocker processes, and what queries are they executing? • Are any deadlocks happening?
Database Activity and Space Monitoring	• What databases are hosted on the SQL server? • Is any of the databases reaching capacity? • Which of the databases is seeing more transaction activity? • How many active transactions are currently happening to each of the database server?
SQL Memory Monitoring	• Is there sufficient memory available for the SQL server? • How much memory is the server consuming and how much is it willing to consume? • How much memory is used for connections, how much for locks, and how much for query optimizations? • What is the server's cache hit ratio? • How many pages are available in the server's buffer pool? • How many of these are free pages?
Operating System Monitoring	• Is there sufficient disk capacity? • Is there excessive contention for CPU or memory resources? • Are the disks unusually busy? • Which processes are taking up most resources (CPU, memory, disk, etc.)?

The **Operating System**, **Network**, **Tcp**, **Application Processes**, and **Windows Service** layers of the layer model in Figure 3.1 have been discussed in the the *Monitoring Unix and Windows Servers* document. Above the Windows Service layer is the **MS SQL Server** layer.

3.1 The MS SQL Server Layer

The tests associated with this layer, indicate the following:

- Error frequency
- SQL engine performance
- SQL Uptime
- Buffer pool usage (specific to MS SQL 2005 and above)
- Active transactions and their effect on tempdb (specific to MS SQL 2005 and above)
- Blocked processes and how to deal with them
- The number and type of system processes currently running

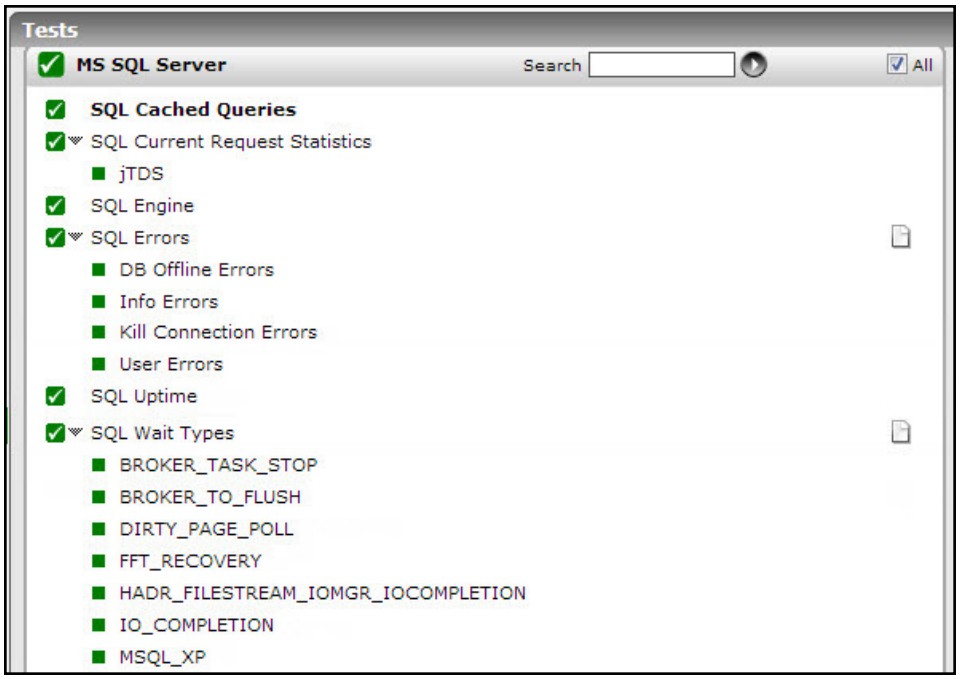


Figure 3.2: The tests associated with the MS SQL Server layer

3.1.1 SQL Wait Types Test

In SQL Server, wait types represent the discrete steps in query processing, where a query waits for resources as the instance completes the request. By analysing wait types and their wait times, administrators can receive quick and objective evidence of performance bottlenecks and their probable causes. The **SQL Wait Types** test enables this analysis. For every type of wait that is currently experienced by the server, this test reports the number, nature, and duration of waits, thereby leading you to the specific wait types that may have contributed to a general slowdown/deterioration in server performance.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every type of wait in the Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the *Sysadmin* role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the connect sql, view server state, view any definition, view any database, and public roles in this text box.
7. **PASSWORD** - The password of the specified **USER**.
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Waiting tasks count:	Indicates the number of	Number	This counter is incremented at the start

Measurement	Description	Measurement Unit	Interpretation
	waits of this type during the last measurement period.		of each wait.
Tasks avg wait time:	Indicates the total wait time for this wait type during the last measurement period.	MilliSecs	A low value is desired for this measure. When a user complaints that query execution takes too long, you can compare the value of this measure across wait types to know which type of wait is the key contributor to delays in query processing.
Tasks avg signal wait time:	Indicates the total signal wait time for this wait type during the last measurement period.	MilliSecs	The signal wait is the time between when a worker has been granted access to the resource and the time it gets scheduled on the CPU. A high value for this measure may imply a high CPU contention. To know which wait type registered the highest signal wait time and probably caused the CPU contention, compare the value of this measure across wait types.
Tasks avg resource wait time:	Indicates the total resource wait time for this wait type during the last measurement period.	MilliSecs	Resource wait time is the actual time a worker waited for the resource to be available. A high value for this measure indicates a delay in acquiring a resource. To know which wait type waited the longest for a resource and therefore contributed to a server slowdown, compare the value of this measure across wait types.
Tasks wait time:	Indicates the percentage of total wait time (across wait types) during which wait events of this type occurred.	Percent	When a user complaints that query execution takes too long, you can compare the value of this measure across wait types to know which type of wait is the key contributor to delays in query processing.

Measurement	Description	Measurement Unit	Interpretation
Tasks signal wait time:	Indicates the percentage of total signal wait time (across wait types) during which wait events of this type waited for a signal.	Percent	The signal wait is the time between when a worker has been granted access to the resource and the time it gets scheduled on the CPU. A high value for this measure may imply a high CPU contention. To know which wait type registered the highest signal wait time and probably caused the CPU contention, compare the value of this measure across wait types.
Tasks resource wait time:	Indicates the percentage of total resource wait time (across wait types) during which wait events of this type waited for a resource.	Percent	Resource wait time is the actual time a worker waited for the resource to be available. A high value for this measure indicates a delay in acquiring a resource. To know which wait type waited the longest for a resource and therefore contributed to a server slowdown, compare the value of this measure across wait types.

3.1.2 SQL Engine Test

The SQL Engine test reports statistics related to the Microsoft SQL server engine.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be

monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.

6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **USE SP MONITOR** - By default, this flag is set to **Yes**, indicating that this test uses the *sp_monitor* stored procedure (by default) to pull out the required metrics from the target server. This stored procedure mandates the *Sysadmin* role - i.e., you should configure the test with the credentials of a **USER** with the *Sysadmin* role, if you want the test to use the *sp_monitor*. Moreover, even if the required privileges are granted to the test, in some environments, the *sp_monitor* procedure may result in errors. Administrators of high-security Windows environments may not want to expose the credentials of their *Sysadmin* users. Neither would they want error-prone stored procedures to execute in their environment. In such environments therefore, you can use queries to extract the desired metrics from the Microsoft SQL server, instead of the *sp_monitor* procedure. To enable the use of queries, set this flag to **No**.
12. **ISPASSIVE** – If the value chosen is **Yes**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cpu usage:	The percentage of time for which the server's CPU was engaged in processing	Percent	A high value of this measure indicates a heavy load on the server. If this value comes close to 100%, it could indicate a

Measurement	Description	Measurement Unit	Interpretation
	requests to the server		probable delay in the processing of subsequent requests to the server. The detailed diagnosis measures associated with the Background processes measure of the MsSqlSysProcesses test will help you identify the processes that are consuming excessive CPU resources.
I/O usage:	The percentage of time for which the server was engaged in performing input/output operations	Percent	
CPU idle time:	The percentage of time for which the server was idle	Percent	A low value of this measure is indicative of high CPU utilization.
Packets received:	The rate at which input packets were read by the SQL server	Pkts/Sec	This measure is an indicator of the traffic to the server.
Packets sent:	The rate at which output packets were read by the SQL server	Pkts/Sec	This measure is an indicator of traffic from the server.
Packet errors:	The rate at which packet errors occurred	Errors/Sec	Ideally, this value should be 0.
Disk reads:	The rate of disk reads performed by the Microsoft SQL server	Reads/Sec	The value of this measure should be kept at a minimum, as disk reads are expensive operations. Ideally, data reads should be performed from the server cache and not directly from the disk. To ensure effective cache usage, allocate adequate memory to the Microsoft SQL server.
Disk writes:	The rate of disk writes performed by the Microsoft SQL server	Writes/Sec	The value of this measure should be kept at a minimum, as disk writes are expensive operations. Ideally, data

Measurement	Description	Measurement Unit	Interpretation
			should be written to the data cache and not directly to the disk. To ensure effective cache usage, allocate adequate memory to the Microsoft SQL server.
Disk I/O errors:	The rate of errors encountered by the Microsoft SQL server while reading and writing	Errors/Sec	Disk read/write errors are normally caused by the following reasons: • Semaphore contention • Excessive disk space consumption

3.1.3 SQL Errors Test

This test reports the rate at which errors occur on the Microsoft SQL Server 2005 (or above). This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every error type on the Microsoft SQL Server 2005 (or above) that is being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
5. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection.

However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.

6. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
7. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target Microsoft SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled Microsoft SQL server, set the **SSL** flag to **Yes**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Error rate:	Indicates the rate at which errors of this type occur on the Microsoft SQL Server 2005 (or above).	Errors/Sec	A very high value of this measure indicates a problem condition requiring further investigation.

3.1.4 SQL Uptime Test

In most production environments, it is essential to monitor the uptime of critical database instances in the infrastructure. By tracking the uptime of each of the database instances, administrators can determine what percentage of time a database instance has been up. Comparing this value with service level targets, administrators can determine the most trouble-prone areas of the infrastructure.

In some environments, administrators may schedule periodic reboots of their database instance. By knowing that a specific database instance has been up for an unusually long time, an administrator may come to know that the scheduled reboot task is not working on a database instance.

This **SQL Uptime** test monitors the uptime of the target Microsoft SQL database instance.

Note:

This test is applicable only to Microsoft SQL Server version 2008 and above.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.

3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the *Sysadmin* role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the connect sql, view server state, view any definition, view any database, and public roles in this text box.
7. **PASSWORD** - The password of the specified **USER**.
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
12. **REPORTMANAGERTIME** – By default, this flag is set to **Yes**, indicating that, by default, the detailed diagnosis of this test, if enabled, will report the shutdown and reboot times of the device in the manager's time zone. If this flag is set to **No**, then the shutdown and reboot times are shown in the time zone of the system where the agent is running(i.e., the system being managed for agent-based monitoring, and the system on which the remote agent is running - for agentless monitoring).
13. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability

- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Has SQL server been restarted?:	Indicates whether the database instance has been rebooted during the last measurement period or not.	Yes	If the value of this measure is *Yes*, it means that the database instance was rebooted during the last measurement period. By checking the time periods when this metric changes from *No* to *Yes*, an administrator can determine the times when this database instance was rebooted. The Detailed Diagnosis of this measure, if enabled, lists the *TIME, SHUTDOWN DATE, RESTART DATE, SHUTDOWN DURATION, and IS MAINTENANCE*.
Uptime since the last measurement:	Indicates the time period that the database instance has been up since the last time this test ran.	Secs	If the database instance has not been rebooted during the last measurement period and the agent has been running continuously, this value will be equal to the measurement period. If the database instance was rebooted during the last measurement period, this value will be less than the measurement period of the test. For example, if the measurement period is 300 secs, and if the database instance was rebooted 120 secs back, this metric will report a value of 120 seconds. The accuracy of this metric is dependent on the measurement period – the smaller the measurement period, greater the accuracy.
Uptime:	Indicates the total time that the database instance has been up since its last reboot.	Mins	This measure displays the number of years, months, days, hours, minutes and seconds since the last reboot.

Measurement	Description	Measurement Unit	Interpretation
			Administrators may wish to be alerted if the database instance has been running without a reboot for a very long period. Setting a threshold for this metric allows administrators to determine such conditions.

3.1.5 SQL System Processes Test

This test reports details about the system processes running on an Microsoft SQL server.

Target of the test : An Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access

the monitored SQL server.

10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

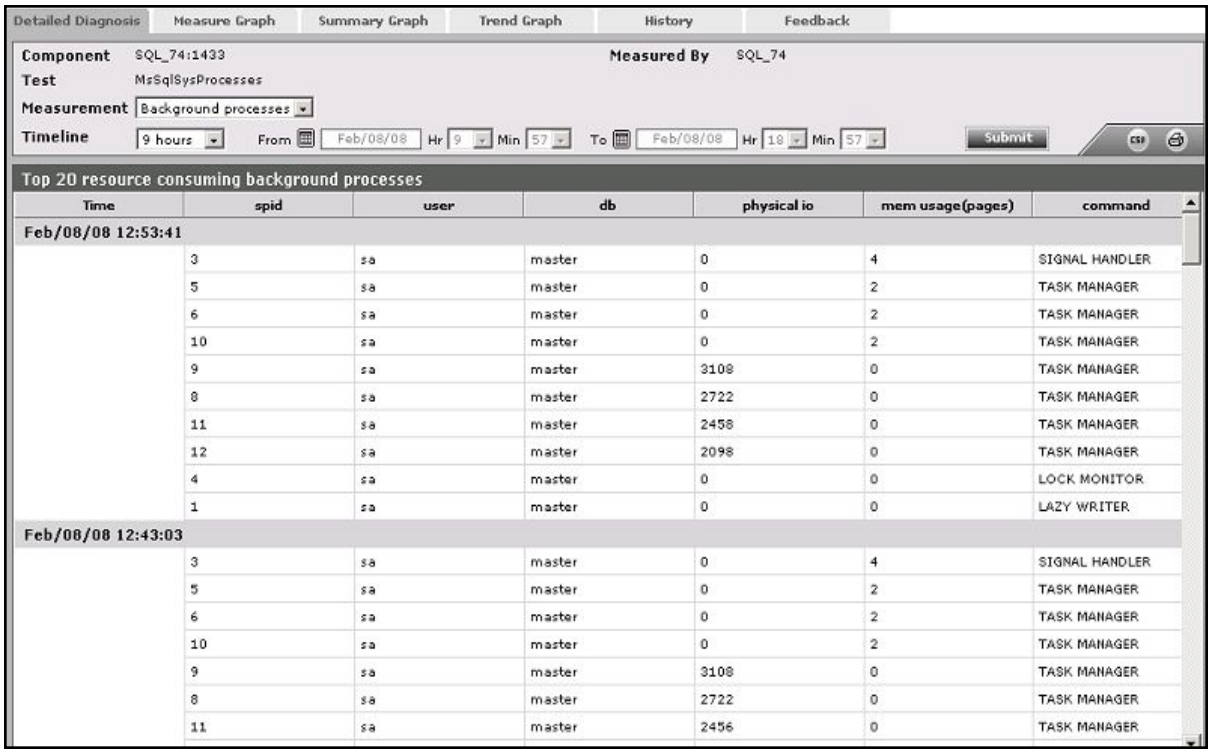
Measurement	Description	Measurement Unit	Interpretation
Total processes:	The total number of Microsoft SQL server processes	Number	The value of this measure is the sum of the number of background, running, sleeping, rollback, and suspended processes.
Background processes:	The total number of background processes run by the Microsoft SQL server rather than by a user process	Number	The detailed diagnosis of this measure, if enabled, provides the details pertaining to the background processes currently executing.
Running processes:	The total number of running processes	Number	The detailed diagnosis of this measure, if enabled, provides details such as the ID of the running processes, the user executing each of the processes, the database on which every process is

Measurement	Description	Measurement Unit	Interpretation
			executing etc. Note that while the query used by the eG agent for collecting the metrics of this test will be counted as a Running process, the detailed diagnosis of this measure will not include this eG query.
Sleeping processes:	The total number of sleeping processes	Number	The detailed diagnosis of this measure, if enabled, provides details such as the ID of the sleeping processes, the user executing each of the processes, the database on which every process is executing, the sleep status, sleep time etc.
Rollback processes:	The total number of processes that were rolled back	Number	The detailed diagnosis of this measure, if enabled, reveals information such as the ID of the rolled back processes, the user executing each of the processes, the database on which every process is executing, etc.
Blocked processes:	If a process attempts to access a resource that is already in use by another process, then such a process will be blocked until such time that the other process releases the resource. This measures indicates the total number of blocked processes.	Number	The detailed diagnosis of the Blocked processes measure, if enabled, reveals information such as the ID of the blocked processes, the user executing each of the processes, the database on which every process is executing, the waiting time of the blocked process, etc. These details aid the user in identifying the blocked processes, the processes that are blocking them (i.e. the process that currently holds a lock on the resource), and also the duration for which the processes have been blocked. If a process is found to hold a lock for too long a time, then such processes can be killed so as to free the resource for the corresponding blocked process.

Measurement	Description	Measurement Unit	Interpretation
Suspended processes:	Indicates the number of processes that are currently suspended.	Number	A Microsoft SQL server marks a process as "suspended" when the process has made a request to a non-SQL process or resource and is awaiting a response. This happens a lot when you have slow disk drives; processes will be suspended while the SQL server waits for the drive to return data or report back after committing. Ideally, the value of this measure should be low. The detailed diagnosis of this measure, if enabled, will provide the complete details of the suspended processes.
	Rollback processes: Indicates the number of sessions initiated by this user in which transaction rollbacks are in progress.	Number	Ideally, the value of this measure should be low. If this value is very close to the Total processes value for a user, it indicates that many transactions executed by that user are being rolled back. This is a cause for concern, as rollbacks are expensive operations that need to be kept at a minimum; if not, processing overheads increase and the overall performance of the server deteriorates. The detailed diagnosis of this measure, if enabled, will provide the complete details of the user sessions with transaction rollbacks.
Dormant processes:	Indicates the number of processes being reset by the Microsoft SQL server.	Number	Ideally, the value of this measure should be low. If this value is high, it indicates that many processes are being reset. The detailed diagnosis of this measure, if enabled, will provide the complete details of the reset processes.

Measurement	Description	Measurement Unit	Interpretation
Pending processes:	Indicates the number of processes that are waiting for a worker thread to become available.	Number	A low value is desired for this measure. If the value of this measure is high, it indicates that many are unable to execute owing to the lack of worker threads. Use the detailed diagnosis of this measure to know which processes are waiting for worker threads.
Spinloop processes:	Indicates the number of processes that are waiting for a spinlock to free.	Number	Spinlocks are lightweight synchronization primitives which are used to protect access to data structures. They are generally used when it is expected that access to a given data structure will need to be held for a very short period of time. When a thread attempting to acquire a spinlock is unable to obtain access it executes in a loop periodically checking to determine if the resource is available instead of immediately yielding. After some period of time a thread waiting on a spinlock will yield before it is able to acquire the resource in order to allow other threads running on the same CPU to execute. This is known as a backoff. The detailed diagnosis of this measure will reveal the processes that are waiting for spinlock to free.

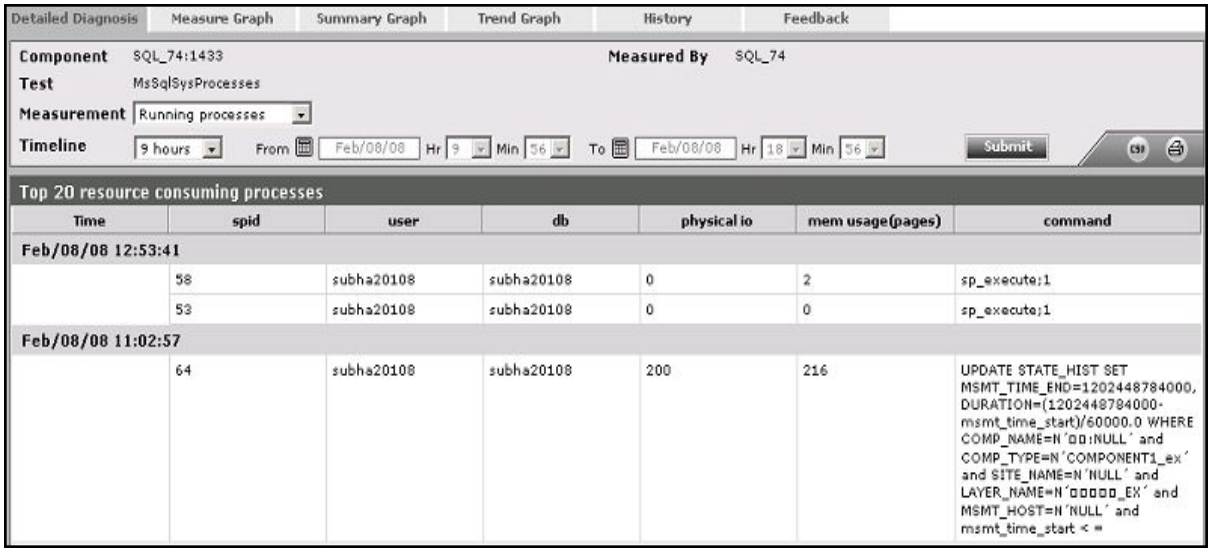
The detailed diagnosis of the *Background processes* measure, if enabled, provides the details pertaining to the background processes currently executing (see Figure 3.3). This information helps the user identify the processes consuming excessive CPU and memory resources. If found necessary, such processes can be killed so as to free adequate CPU resources.



Time	spid	user	db	physical io	mem usage(pages)	command
Feb/08/08 12:53:41						
	3	sa	master	0	4	SIGNAL HANDLER
	5	sa	master	0	2	TASK MANAGER
	6	sa	master	0	2	TASK MANAGER
	10	sa	master	0	2	TASK MANAGER
	9	sa	master	3108	0	TASK MANAGER
	8	sa	master	2722	0	TASK MANAGER
	11	sa	master	2458	0	TASK MANAGER
	12	sa	master	2098	0	TASK MANAGER
	4	sa	master	0	0	LOCK MONITOR
	1	sa	master	0	0	LAZY WRITER
Feb/08/08 12:43:03						
	3	sa	master	0	4	SIGNAL HANDLER
	5	sa	master	0	2	TASK MANAGER
	6	sa	master	0	2	TASK MANAGER
	10	sa	master	0	2	TASK MANAGER
	9	sa	master	3108	0	TASK MANAGER
	8	sa	master	2722	0	TASK MANAGER
	11	sa	master	2456	0	TASK MANAGER

Figure 3.3: The detailed diagnosis of the Background processes measure

The detailed diagnosis of the *Running processes* measure, if enabled, provides details such as the ID of the running processes, the user executing each of the processes, the database on which every process is executing etc. This information enables the user to understand the general user behavior on the server (see Figure 3.4).



Time	spid	user	db	physical io	mem usage(pages)	command
Feb/08/08 12:53:41						
	58	subha20108	subha20108	0	2	sp_execute;1
	53	subha20108	subha20108	0	0	sp_execute;1
Feb/08/08 11:02:57						
	64	subha20108	subha20108	200	216	UPDATE STATE_HIST SET MSMT_TIME_END=1202448784000, DURATION=(1202448784000-msmt_time_start)/60000.0 WHERE COMP_NAME=N'□□:NULL' and COMP_TYPE=N'COMPONENT1_ex' and SITE_NAME=N'NULL' and LAYER_NAME=N'□□□□□_EX' and MSMT_HOST=N'NULL' and msmt_time_start < =

Figure 3.4: The detailed diagnosis of the Running processes measure

The detailed diagnosis of the *Sleeping processes* measure, if enabled, provides details such as the ID of the sleeping processes, the user executing each of the processes, the database on which every process is executing, the sleep status, sleep time etc. Using this information, users can identify those processes that have been idle for a long period of time (see Figure 3.5).

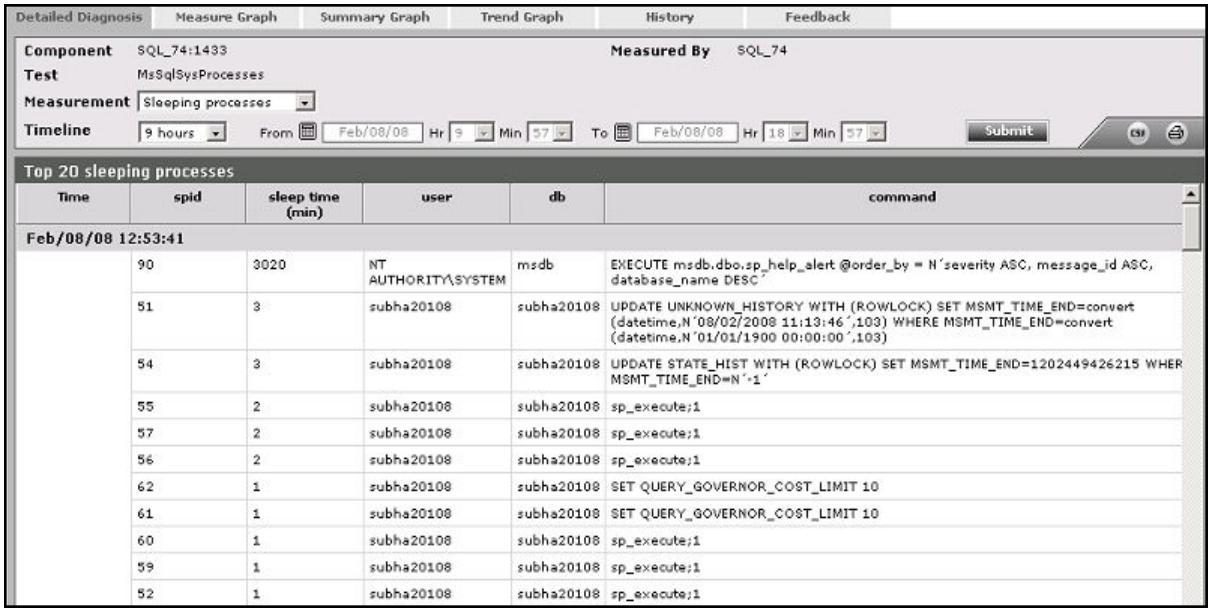


Figure 3.5: The detailed diagnosis of the Sleeping processes measure

The detailed diagnosis of the *Rollback processes* measure, if enabled, reveals information such as the ID of the rolledback processes, the user executing each of the processes, the database on which every process is executing, etc. Rollbacks are expensive operations on a server. The detailed measures provided by eG in this regard, enable the user to isolate the specific queries that have rolledback. Further analysis of these queries can be performed, in order to figure out the reason for the rollback and take adequate measures to prevent it from recurring.

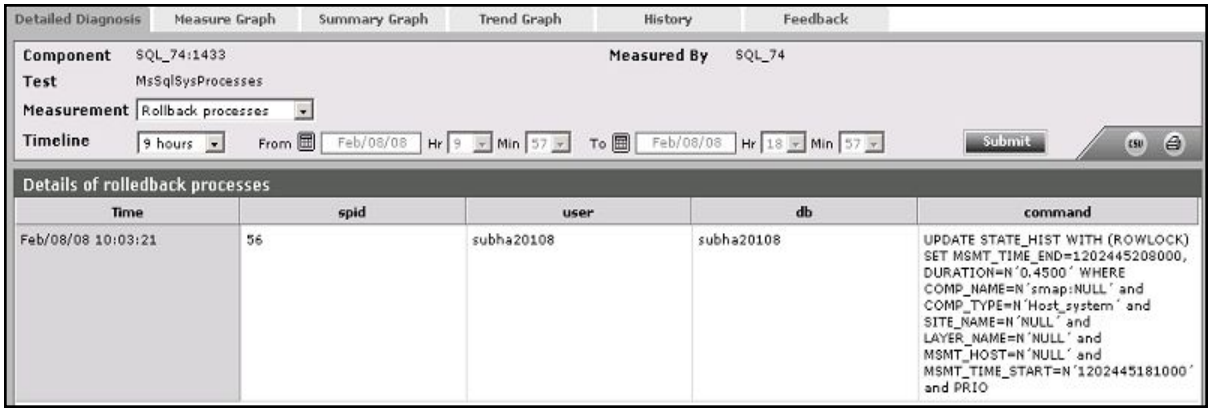


Figure 3.6: The detailed diagnosis of the Rollback processes measure

3.1.6 SQL Current Request Statistics Test

In the database context, the connection between the user process and the server process is called a session. The server process communicates with the connected user process and performs tasks on behalf of the users.

This test tracks the resource usage of the sessions to the target Microsoft SQL server. In the process, the test turns the spotlight on resource-intensive SQL server sessions and the queries executed by such sessions that may require fine-tuning. Additionally, the test also reports the average wait time of sessions, leads you to that session that has been waiting for the maximum time, and points you to the exact query that the session has been taking too long to execute. Inefficient queries are thus revealed, enabling you to quickly initiate query optimization measures.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the *Sysadmin* role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the connect sql, view server state, view any definition, view any database, and public roles in this text box.
7. **PASSWORD** - The password of the specified **USER**.
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of

Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Avg memory usage:	Indicates the average amount of memory that is currently used by all sessions of this Microsoft SQL server.	KB	A high value indicates that one/more Microsoft SQL sessions are consuming high memory. Use the detailed diagnosis of the Max memory usage measure to identify which session is consuming maximum memory. To reduce the memory consumption of the session, you may have to optimize the query displayed in the SQL TEXT column of the detailed diagnosis. To optimize the query, you would be required to do any one of the following: • Check the fragmentation activity of the disks of the Microsoft SQL server; • Add additional indexes to the Microsoft SQL server or;

Measurement	Description	Measurement Unit	Interpretation
			• Include a hint to the query which would considerably reduce the memory usage of the server.
Max memory usage:	Indicates the maximum memory used by the SQL server sessions.	KB	The detailed diagnosis of this measure, if enabled, displays the session ID, the name of the database accessed by the session, the login name of the user who initiated the session, the login time of the user, the request start time, when the session was established, the session wait time and type, the session duration, the time for which the session hogged the CPU, the memory usage of the session, the session status, the total number of reads, writes, and logical reads performed by the session on the database, and the query executed by the session. From this information, you can easily identify the session that is consuming the maximum CPU/memory, the session that has been waiting for the maximum time for the query to execute, the session that has performed the maximum I/O activities on the SQL server, and the query that is responsible for all such resource-intensive tasks.
Avg CPU time:	Indicates the average time for which the SQL sessions used the CPU resources of the SQL server.	Secs	A high value indicates that one/more Microsoft SQL sessions are hogging the CPU. Use the detailed diagnosis of the Max CPU time measure to identify which session is consuming the CPU resources excessively. To reduce the CPU consumption of a session, you may have to optimize the query displayed in the SQL TEXT column of the detailed diagnosis. To optimize the query, you would be

Measurement	Description	Measurement Unit	Interpretation
			required to do any one of the following: • Check the fragmentation activity of the disks of the Microsoft SQL server; • Add additional indexes to the Microsoft SQL server or; • Include a hint to the query which would considerably reduce the memory usage of the server.
Max CPU time:	Indicates the maximum time for which the SQL sessions used the CPU.	Secs	The detailed diagnosis of this measure, if enabled, displays the session ID, the name of the database accessed by the session, the login name of the user who initiated the session, the login time of the user, the request start time, when the session was established, the session wait time and type, the session duration, the time for which the session hogged the CPU, the memory usage of the session, the session status, the total number of reads, writes, and logical reads performed by the session on the database, and the query executed by the session. From this information, you can easily identify the session that is consuming the maximum CPU/memory, the session that has been waiting for the maximum time for the query to execute, the session that has performed the maximum I/O activities on the SQL server, and the query that is responsible for all such resource-intensive tasks.
Avg wait time:	Indicates the average time for which the SQL sessions were waiting.	Secs	A high value indicates that one/more Microsoft SQL sessions are waiting too long to perform a task – typically, to execute a query. Use the detailed

Measurement	Description	Measurement Unit	Interpretation
			diagnosis of the Max wait time measure to identify which session is taking too long for query execution. To reduce the wait time of a session, you may have to optimize the query displayed in the SQL TEXT column of the detailed diagnosis. To optimize the query, you would be required to do any one of the following: • Check the fragmentation activity of the disks of the Microsoft SQL server; • Add additional indexes to the Microsoft SQL server or; • Include a hint to the query which would considerably reduce the memory usage of the server.
Max wait time:	Indicates the maximum time for which the SQL sessions waited.	Secs	The detailed diagnosis of this measure, if enabled, displays the session ID, the name of the database accessed by the session, the login name of the user who initiated the session, the login time of the user, the request start time, when the session was established, the session wait time and type, the session duration, the time for which the session hogged the CPU, the memory usage of the session, the session status, the total number of reads, writes, and logical reads performed by the session on the database, and the query executed by the session. From this information, you can easily identify the session that is consuming the maximum CPU/memory, the session that has been waiting for the maximum time for

Measurement	Description	Measurement Unit	Interpretation
			the query to execute, the session that has performed the maximum I/O activities on the SQL server, and the query that is responsible for all such resource-intensive tasks.
Avg I/O time for current queries:	Indicates the average time taken by current queries for I/O processing.	Secs	A low value is desired for this measure. A high value indicates that one/more queries are I/O-intensive.
Max I/O time for current queries:	Indicates the maximum time that the current queries took for I/O processing.	Secs	If the value of this measure exceeds 10 seconds, you will have to check the disk I/O subsystem for the proper placement of files – LDF and MDF on separate drives, tempDB on a separate drive, hot spot tables on separate filegroups. I/O can also be reduced if the SQL server uses cover index instead of cluster index.

3.1.7 SQL Cached Queries Test

SQL Server maintains a cache, but not with canned results for queries. In an OLTP system, many tables are frequently updated; it is therefore highly unlikely that the same query yields the same result twice. Similarly, the likelihood of the same query reappearing with exactly the same parameters is also very less. What SQL Server stores in its cache therefore, are recently accessed data pages, as well as query plans for recently submitted queries and invoked stored procedures. This makes it possible to retrieve the result of a query without accessing the disk for frequently accessed tables. Too few queries in the cache means more direct disk accesses! To minimize reads/writes to physical disks, more number of queries should execute in the cache. Using this test, you can determine the number of queries that are currently executing in the cache and also figure out the impact of cache misses on the physical disks. These metrics reveal whether/not cache usage is at a desired level. In the process, the test also measures the resource usage of and the I/O activity generated by the cached queries and sheds light on time-consuming, resource-intensive and I/O-intensive queries that are executing in the cache.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the *Sysadmin* role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the connect sql, view server state, view any definition, view any database, and public roles in this text box.
7. **PASSWORD** - The password of the specified **USER**.
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability

- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

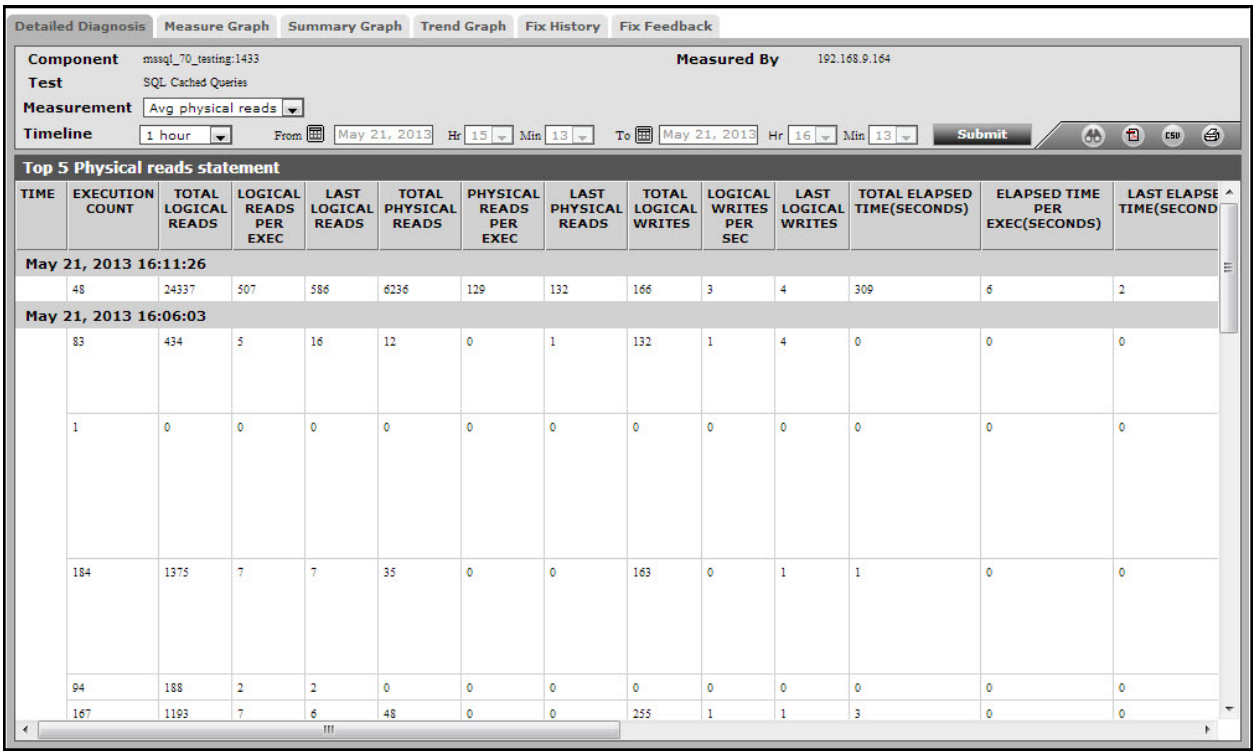
Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Physical reads:	Indicates the rate at which the queries directly executed on and read from the physical disk.	Reads/Sec	A high value could indicate that direct disk accesses are occurring too frequently. This in turn implies poor cache usage. You may consider resizing your cache to accommodate more number of queries, so that direct disk reads are reduced.
Avg physical reads:	Indicates the average number of reads performed by the queries that executed directly on the physical disk.	Number	A high value indicates that one/more queries are reading too frequently from the physical disk. This is an unhealthy practice and can be attributed poor cache usage. Use the detailed diagnosis of this measure to know which queries are not executing in the cache and the number of times each of these queries read directly from the physical disk. This way, you can quickly identify that query which exerts the maximum pressure on the physical disk.
Logical reads:	Indicates the rate of data reads performed by queries to the cache.	Reads/Sec	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size.
Avg logical reads:	Indicates the average number of data reads performed by queries to the cache.	Number	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size. You can also use the detailed diagnosis of this measure to view the top-5

Measurement	Description	Measurement Unit	Interpretation
			queries in terms of number of logical reads. This way, you can precisely identify the most I/O-intensive query to the cache.
Logical writes:	Indicates the rate at which the cached queries performed writes.	Writes/Sec	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size.
Avg logical writes:	Indicates the average number of times data writes were performed by a query to the cache.	Number	A high value is desired for this measure. A low value is indicative of ineffective cache usage, typically caused by improper cache size. You can also use the detailed diagnosis of this measure to view the top-5 queries in terms of number of logical writes. This way, you can precisely identify the most I/O-intensive query to the cache.
CPU time:	Indicates the percentage of time for which the cached queries hogged the CPU.	Percent	A high value is indicative of excessive CPU usage by the cached queries. Use the detailed diagnosis of this measure to know which query is CPU-intensive.
Max elapsed time:	Indicates the maximum time taken by the cached queries for execution.	Secs	If the value of this measure is very high, it could either indicate that the database is unable to process the queries quickly or that one/more queries to the database are taking too long to execute. Improper indexing and fragmented tables in the database are common causes for slowdowns at the database-level. Besides the above, queries that are improperly structured can also take time to execute. The longer a query executes on the database, higher would be the resource consumption of that

Measurement	Description	Measurement Unit	Interpretation
			query. It is therefore imperative that such resource- intensive queries are quickly isolated and fine-tuned, so as to prevent degradations in the performance of the database server. Using the detailed diagnosis of this measure, you can rapidly identify the resource- intensive queries to the database.
Recently executed queries:	Indicates the number of queries that executed in the cache since the last measurement period.	Number	A consistent rise in the value of this measure is a sign of optimal cache usage and minimal direct disk accesses.

The detailed diagnosis of the *Avg physical read* measure lists the top-5 queries in terms of the number of times they read directly from the physical disk. The query that exerts the maximum pressure on the disk can thus be isolated.



TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	48	24337	507	586	6236	129	132	166	3	4	309	6	2
May 21, 2013 16:06:03													
	83	434	5	16	12	0	1	132	1	4	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0
	184	1375	7	7	35	0	0	163	0	1	1	0	0
	94	188	2	2	0	0	0	0	0	0	0	0	0
	167	1193	7	6	48	0	0	255	1	1	3	0	0

Figure 3.7: The detailed diagnosis of the Avg physical reads measure

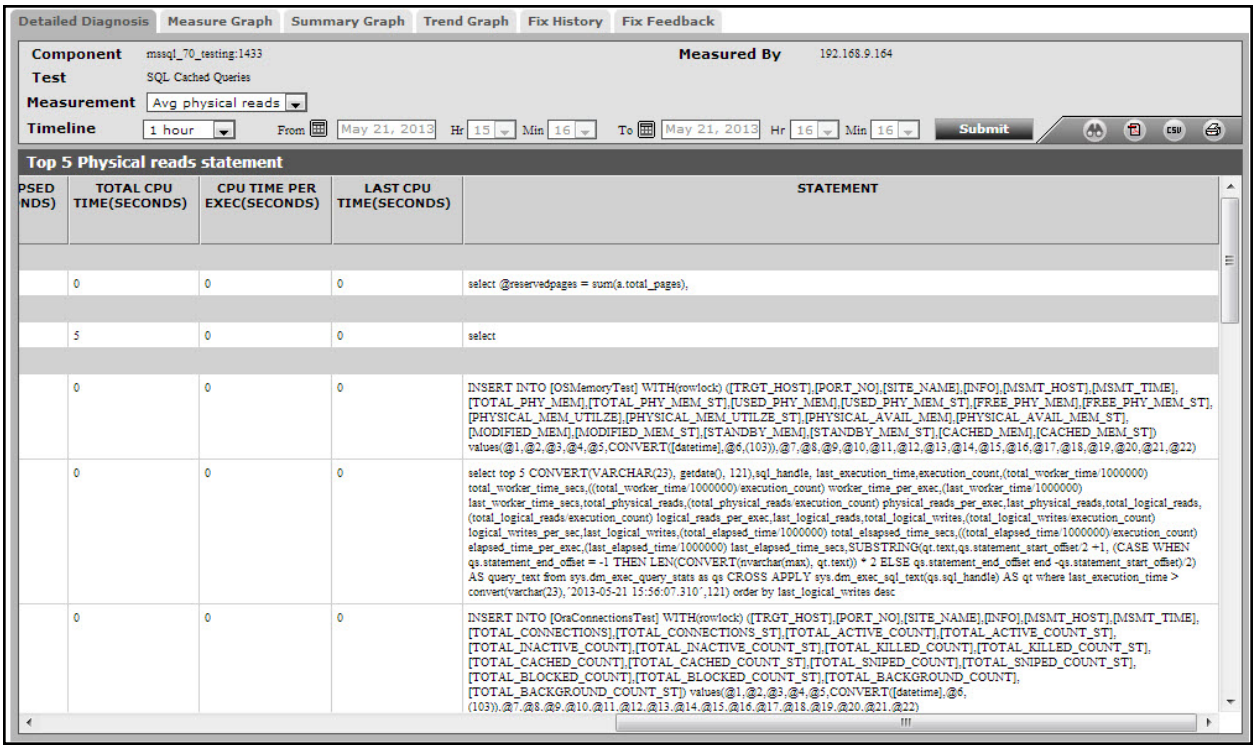


Figure 3.8: The detailed diagnosis of the Avg physical reads measure

The detailed diagnosis of the *Avg logical reads* measure lists the top-5 cached queries in terms of the number of logical reads. The cached query that performed the maximum number of data reads can be identified. You can even analyse the query to figure out whether the number of reads it generates is justified or not; if not, you may have to optimize the query.

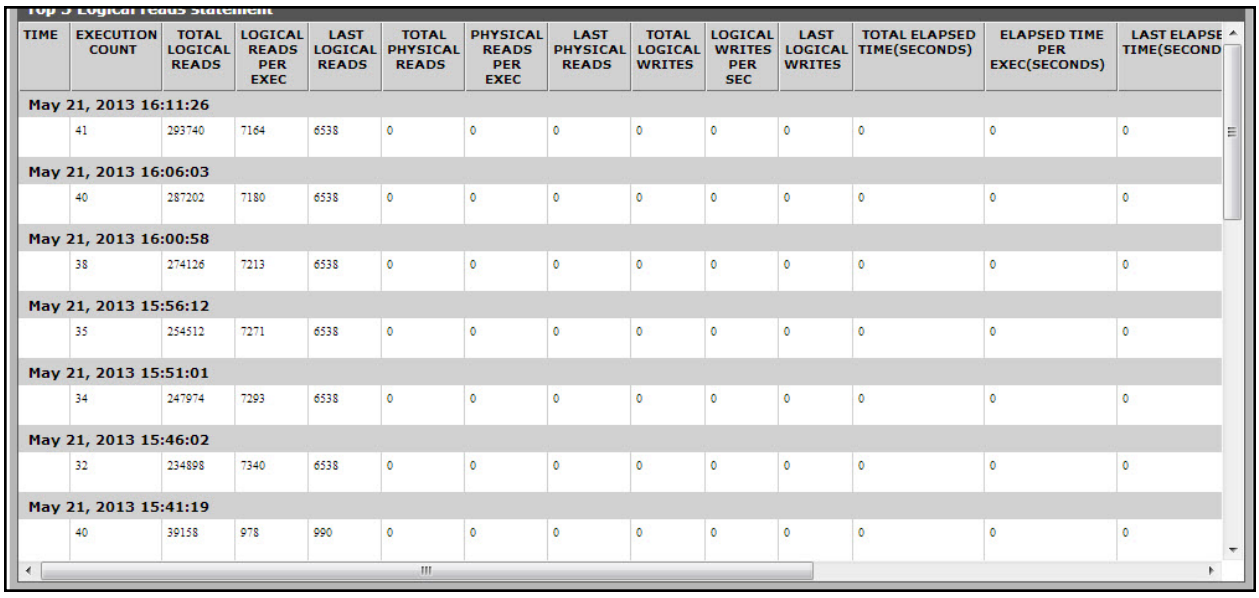
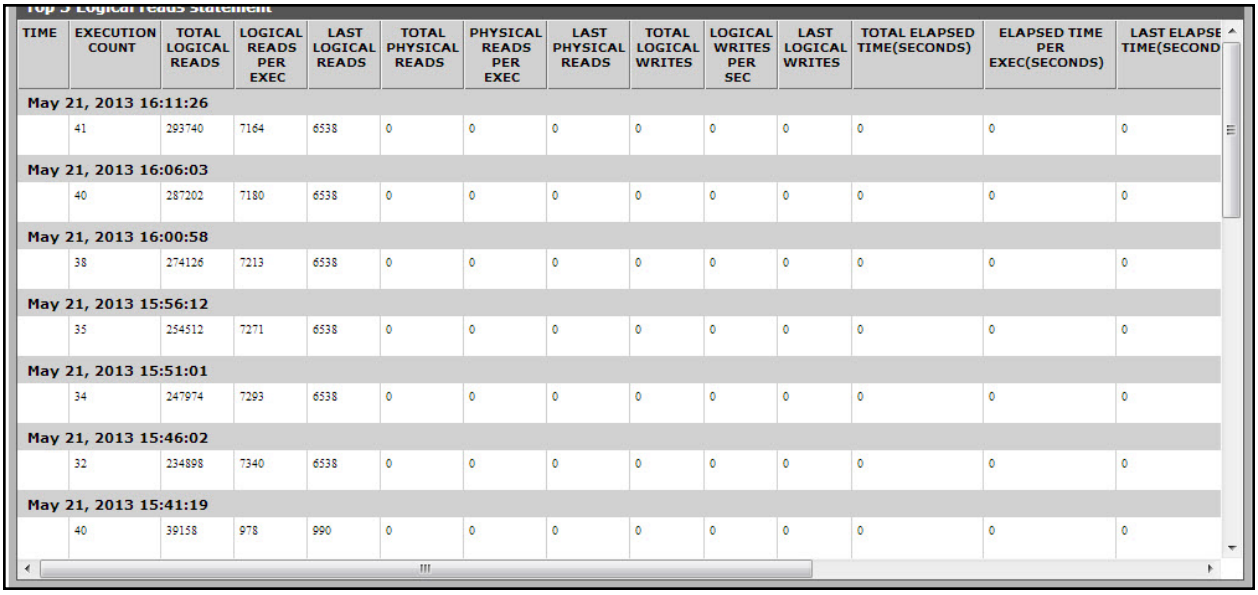


Figure 3.9: The detailed diagnosis of the Avg logical reads measure

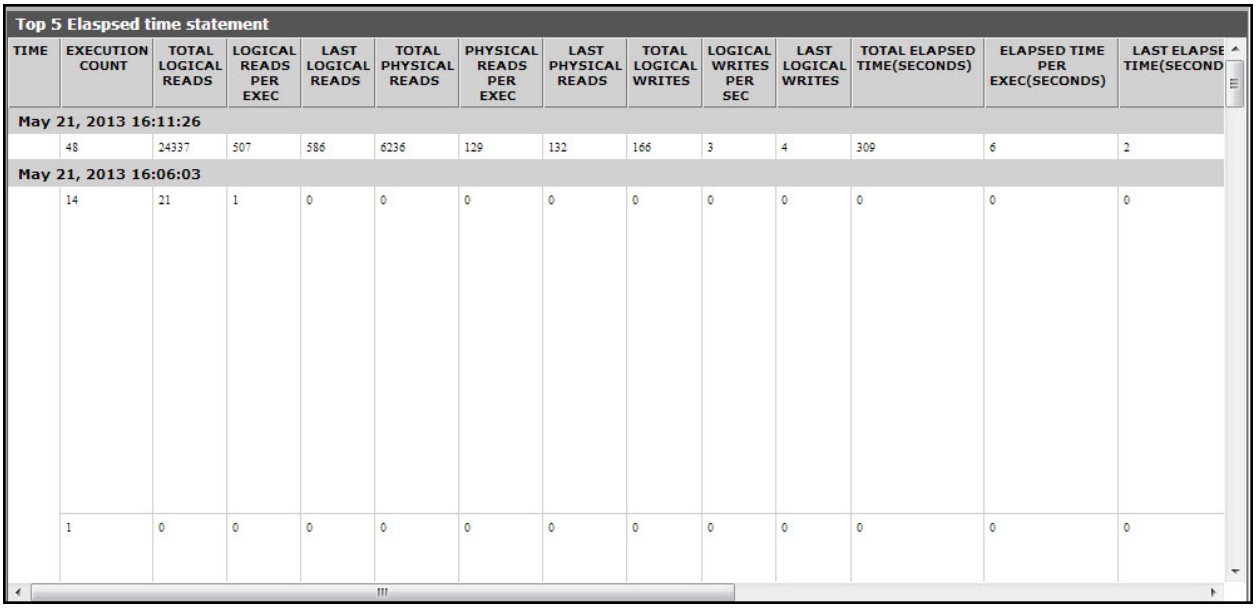
The detailed diagnosis of the *Avg logical writes* measure lists the top-5 cached queries in terms of the number of logical writes. The cached query that performed the maximum number of data writes can be inferred from this. You can even analyze the query to figure out whether the number of writes it generates is justified or not; if not, you may have to optimize the query.



TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	41	293740	7164	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 16:06:03													
	40	287202	7180	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 16:00:58													
	38	274126	7213	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:56:12													
	35	254512	7271	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:51:01													
	34	247974	7293	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:46:02													
	32	234898	7340	6538	0	0	0	0	0	0	0	0	0
May 21, 2013 15:41:19													
	40	39158	978	990	0	0	0	0	0	0	0	0	0

Figure 3.10: The detailed diagnosis of the Avg logical reads measure

The detailed diagnosis of the *Max elapsed time* measure lists the top-5 cached queries in terms of the time they took to complete execution. The query that took the longest time to execute can thus be easily identified. You can analyse whey that query took long to execute, assess the resource foot print of that query, and if required, attempt to fine-tune the query to reduce execution time / resource usage.



Top 5 Elaspsed time statement

TIME	EXECUTION COUNT	TOTAL LOGICAL READS	LOGICAL READS PER EXEC	LAST LOGICAL READS	TOTAL PHYSICAL READS	PHYSICAL READS PER EXEC	LAST PHYSICAL READS	TOTAL LOGICAL WRITES	LOGICAL WRITES PER SEC	LAST LOGICAL WRITES	TOTAL ELAPSED TIME(SECONDS)	ELAPSED TIME PER EXEC(SECONDS)	LAST ELAPSE TIME(SECOND
May 21, 2013 16:11:26													
	48	24337	507	586	6236	129	132	166	3	4	309	6	2
May 21, 2013 16:06:03													
	14	21	1	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.11: The detailed diagnosis of the Max elapsed time measure

The detailed diagnosis of the *Cpu time* measure lists the top-5 cached queries in terms of CPU usage. The most CPU-intensive query can thus be identified and the reasons for the same can be determined.

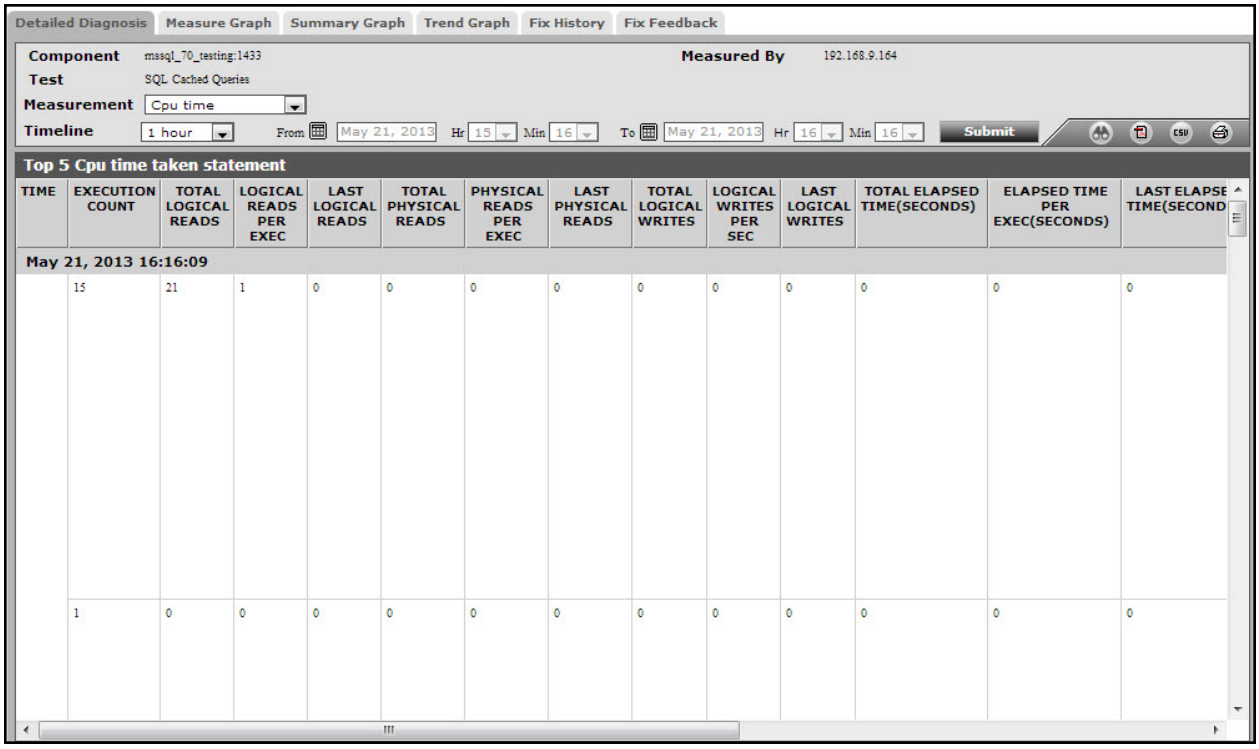


Figure 3.12: The detailed diagnosis of the Cpu time measure

3.1.8 SQL AlwaysOn Availability Test

The AlwaysOn Availability Groups feature is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. Introduced in SQL Server 2012, AlwaysOn Availability Groups maximizes the availability of a set of user databases for an enterprise. An *availability group* supports a failover environment for a discrete set of user databases, known as *availability databases*, that failover together. An availability group supports a set of read-write primary databases and one to eight sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations. An availability group fails over at the level of an availability replica. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file, deletion of a database, or corruption of a transaction log. In environments where critical data is stored, it is important for the database to be highly available. The AlwaysOn Availability Groups if enabled, will help administrators in maintaining such high availability. Therefore, it is important to monitor the status of the AlwaysOn Availability Groups. The **SQL AlwaysOn Availability** test exactly helps in this regard.

This test reports whether the AlwaysOn Availability Group feature is enabled and the current state of the AlwaysOn Availability Groups manager.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the MS SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the MS SQL server.
3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433.
4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER.**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Is always- on enabled?:	Indicates whether/not the Always on Availability Groups feature is enabled.		The values reported by this measure and their numeric equivalents are available in the table below:

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>No</td><td>0</td></tr>
<tr><td>Yes</td><td>1</td></tr>
</table>
Note:
This measure reports the Measure Values listed in the table above to indicate whether/not the Always on Availability Groups concept was enabled. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
</td></tr>
<tr><td>Alwayson manager status:</td><td>Indicates the current state of the Always on Availability Groups manager.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>Failed</td><td>0</td></tr>
<tr><td>Pending Communication</td><td>1</td></tr>
<tr><td>Running</td><td>2</td></tr>
</table>
Note:
This measure reports the Measure Values listed in the table above to indicate the current status of the Always on Availability Groups manager. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
</td></tr>
</table>

3.1.9 SQL AlwaysOn Member Status Test

The AlwaysOn feature not only combines the power of clustering and mirroring into one High Availability option, but also allows you to interact with the secondary databases. In addition, AlwaysOn Availability Groups allows you to configure failover for one database, a set of databases or the entire instance. Another important aspect of the AlwaysOn is that you can create multiple failover targets. If the Availability Group is enabled on multiple Microsoft SQL server instances in a cluster, then the administrators are required to monitor each member of the cluster node that is enabled with the AlwaysOn feature. In addition, if the failover

concept has to be fool-proof then there arises a need for a file-share witness or a disk witness. The file-share witness or disk witness is most commonly used when shared storage is available to a cluster. The file share witness or disk witness pings the members of a cluster and syncs the data from all the members to keep the database updated. In case of failover, the disk witness or file share witness will render the cluster node with an updated database. Since the file share witness or disk witness stores the updated database by constantly pinging the members of the cluster node on which AlwaysOn feature is enabled, it becomes mandatory to check the status of each member of the cluster node and the disk witness or file share witness. The **SQL AlwaysOn Member Status** test exactly helps you in this regard.

For each category available in the Microsoft SQL server instance enabled with AlwaysOn feature, this test reports the current status of each member. If the member is online, then this test will report whether/not the member is a primary member and whether the member has failed over.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each Category:Member available in the Microsoft SQL server that is being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the MS SQL server.
3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433.
4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an MS SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER.**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the

ISNTLMV2 flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Member status:	Indicates the current state of this member.		The values reported by this measure and their numeric equivalents are available in the table below: **Measure Value** / **Numeric Value** Offline – 0 Online – 1 **Note:** This measure reports the **Measure Value**s listed in the table above to indicate current state of this member. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
Is primary?:	Indicates whether/not this member is the primary member.		The values reported by this measure and their numeric equivalents are available in the table below: **Measure Value** / **Numeric Value** No – 0 Yes – 1 **Note:** This measure reports the **Measure Value**s listed in the table above to indicate whether/not this member is the primary member. However, in the graph,

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>this measure is indicated using the Numeric Values listed in the above table.

This measure is not applicable for the DISK_WITNESS descriptor.</td></tr>
<tr><td>Is switch over happened?:</td><td>Indicates whether/not this member has failed over i.e., this member has switched over from primary to secondary and vice versa.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>No</td><td>0</td></tr><tr><td>Yes</td><td>1</td></tr></table>
Note:

This measure reports the Measure Values listed in the table above to indicate whether/not this member has failed over. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.

This measure is not applicable for the DISK_WITNESS descriptor.</td></tr>
</table>

3.1.10 SQL AlwaysOn Network Latency Test

If transaction log records are not sent quickly by the primary database or are not applied quickly by the secondary database, then the data in the primary and secondary databases will be out of sync; this will cause significant data loss during a failover. To avoid this, administrators must keep track of the log record traffic between the primary and secondary databases, proactively detect potential slowness in synchronization, figure out the probable source of the bottleneck, and clear it to ensure proper synchronization between the primary and secondary databases. This is where the **SQL AlwaysOn Network Latency** test helps.

This test measures the rate at which transaction log data is sent to the secondary database for synchronization on each SQL server instance, and the time taken by the secondary database to apply the data. In the process, the test pinpoints bottlenecks in database synchronization and where exactly the bottlenecks lie.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the *Sysadmin* role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the *connect sql, view server state, view any definition, view any database*, and *public roles* in this text box.
7. **PASSWORD** - The password of the specified **USER**.
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Log sent:	Indicates the amount of data (in bytes) sent from the primary availability replica to the secondary availability replica per second during the last measurement period.	KB/sec	
Log transport:	Indicates the amount of data (in bytes) sent over the network from the primary availability replica to the secondary availability replica per second during the last measurement period.	KB/sec	
Log send wait time:	Indicates the time duration for which the log stream messages were waiting in the Flow Control mode per second.	Msecs/sec	Ideally, the value of this measure should be low. A gradual/sudden increase in this measure indicates that the network over which the log messages are sent is experiencing slowdowns/network delays and noise. A high value for this measure is also indicative of potential data loss which is much more than the estimated Recovery Point Objective (RPO).
Log send waits:	Indicates the number of times Flow Control mode was initiated per second.	Waits/sec	A high value for this measure indicates that the network is congested and is experiencing slowdowns.
Avg log send wait time:	Indicates the average time the log messages should wait in the Flow Control mode.	Secs/Wait	This measure is a ratio of the Log send wait time and the *Log send waits* measures. A low value is desired for this measure.
Alwayson messages	Indicates the number of	Number	Ideally, the value of this measure should

Measurement	Description	Measurement Unit	Interpretation
resent:	Always on messages i.e., log stream messages that were resent over the network during the last measurement period.		be low. A high value for this measure is a cause of concern as this indicates a high network latency or network congestion or network noise.

3.1.11 SQL AlwaysOn Page Repair Test

The AlwaysOn Availability Groups support automatic page repair. After certain types of errors corrupt a page on a local database, making it unreadable, an availability replica (primary or secondary) attempts to automatically recover the page by resolving those errors that prevent reading the data in the page. If a secondary replica cannot read the page, the replica requests a fresh copy of the page from the primary replica. If the primary replica cannot read the page, the replica broadcasts a request for a fresh copy to all the secondary replicas and gets the page from the first replica that responds. If this request succeeds, the unreadable page is replaced by the copy, which usually resolves the error. If the database encounters too many errors simultaneously corrupting a large volume of pages, then the automatic page repair tries to resolve the errors. Certain types of pages such as File header page, Page 9 (the database boot page), the allocation pages etc cannot be repaired by the automatic page repair. If there are too many pages that cannot be repaired, then it indicates that the database is highly corrupted. If the administrators are warned proactively about the number of pages that cannot be repaired, then remedial action can be taken before serious issues occur! The **SQL AlwaysOn Page Repair** test helps administrators to proactively take remedial measures!

For each message returned, this test reports the number of pages on which page repair was attempted. Using the detailed diagnosis of this test, administrators may be able to identify the pages on which page repair was attempted and the pages that cannot be repaired. Using this test, administrators can proactively identify the pages that are corrupted and take remedial measures before the availability databases become unavailable.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each message returned after page repair on the Microsoft SQL Server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the MS SQL server.
3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433.
4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.

5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER.**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Page count:	Indicates the number of pages that returned this message after page repair was attempted.	Number	The detailed diagnosis of this measure if enabled, lists the DatabaseID, FileID, PageID, Error type and Page modification time.

3.1.12 SQL AlwaysOn Recovery Point Test

Recovery Point Objective (RPO) is defined as the amount of acceptable data loss or the point in time up to which the data can be recovered. Whenever a failover is detected, the administrators may want the secondary database to take over quickly from the primary database. If large quantity of data is not transferred to the secondary database from the primary database, then the users have to wait for a longer period to access the databases during failover. Often there would be a minimal data loss when a failover is in progress. This data loss may be due to the time lag that occurs during synchronization that happens between the primary and

secondary databases. If the time taken is too long, it indicates that the synchronization process between the primary and secondary databases is taking too long to complete. This in turn will affect the users who will be compelled to wait for a prolonged time period to access the databases. To avoid such scenarios, it is essential to monitor the recovery point objective of the SQL server. The **SQL AlwaysOn Recovery Point** test helps administrators in this regard.

This test reports the amount of logs that had not been synchronized with the secondary database and the amount of hardened logs that are yet to be applied to the secondary database. In addition, this test helps administrators to analyze the time duration for which the log records were waiting in the redo queue before being rolled to the secondary database. This way, administrators may be proactively alerted to fine tune the time taken to roll the log to the secondary database so that the synchronization process completes in a quick and hassle free manner.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the MS SQL server.
3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433.
4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an MS SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER.**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft

SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Log bytes flushed:	Indicates the rate at which log bytes were flushed to the secondary database to complete synchronization since the last recovery point.	Flushes/sec	If the value of this measure is consistently increasing, then it indicates that the potential data loss can increase indefinitely.
Log send queue size:	Indicates the amount of log that had not been sent to the secondary database from this database to complete synchronization.	KB	Ideally, the value of this measure should be zero. A high value for this measure indicates that this much of data is unavailable in the secondary database during failover which directly implies that the customers would experience this data loss equal to this measure.
Redo queue size:	Indicates the total number of kilobytes of hardened log that currently remain to be applied to the secondary database to roll it forward.	KB	A low value is desired for this measure.
Redo rate:	Indicates the rate at which log records were rolled forward on the secondary database from this database.	KB/sec	
Pending logs recovery time:	Indicates the time duration for which the log records were waiting in the redo queue until being rolled forward to the secondary database.	Secs	Ideally, the value of this measure should be low.

Measurement	Description	Measurement Unit	Interpretation
Pending logs flushed time:	Indicates the time duration for which the logs were in the send queue until being flushed completely to the secondary database.	Secs	

3.1.13 SQL AlwaysOn Replica Status Test

An availability replica is an instantiation of an availability group that is hosted by a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. There are two types of availability replicas that exist in the availability group: single *primary replica* and one to eight *secondary replicas*.

An availability group fails over at the level of an availability replica. An availability replica provides redundancy only at the database level—for the set of databases in one availability group. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file or corruption of a transaction log. The primary replica makes the primary databases available for read-write connections from clients. Also, in a process known as data synchronization, which occurs at the database level. The primary replica sends transaction log records of each primary database to every secondary database. Every secondary replica caches the transaction log records (hardens the log) and then applies them to its corresponding secondary database.

Whenever a failover is detected, the administrators may want the secondary replica to take over quickly from the primary replica. If the primary replica and secondary replicas are not in a position to apply the transaction logs to the primary and secondary databases, then there may be too much of non-sync between the primary replica and the secondary replicas during failover. In order to minimize such synchronization problems and maintain the secondary replicas on par with the primary replica, administrators are required to continuously monitor the operational state, synchronization status and synchronization health of the availability replicas. The **SQL AlwaysOn Replica Status** test helps administrators in this regard! This test continuously monitors the operational state, synchronization state and synchronization health of each availability replica of the SQL AlwaysOn Availability groups. In addition, administrators would be alerted to the current recovery state of the availability replica after a failover is initiated.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each availability replica on the Microsoft SQL server that is being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the MS SQL server.
3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433.
4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific MS SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an MS SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – While monitoring a Microsoft SQL Server 2012 and above, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER.**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Operational state:	Indicates the current operational state of this availability replica.		The values reported by this measure and their numeric equivalents are available in the table below:

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>FAILED_ NO_ QUORUM</td><td>0</td></tr>
<tr><td>FAILED</td><td>1</td></tr>
<tr><td>OFFLINE</td><td>2</td></tr>
<tr><td>PENDING_ FAILOVER</td><td>3</td></tr>
<tr><td>PENDING</td><td>4</td></tr>
<tr><td>ONLINE</td><td>5</td></tr>
</table>
Note:
This measure reports the Measure Values listed in the table above to indicate the states of this availability replica. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
</td></tr>
<tr><td>Recovery state:</td><td>Indicates the current recovery state of this availability replica.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>ONLINE_ IN_ PROGRESS</td><td>0</td></tr>
<tr><td>ONLINE</td><td>1</td></tr>
</table>
Note:
This measure reports the Measure Values listed in the table above to indicate the recovery states of this availability replica. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
</td></tr>
<tr><td>Synchronization health state:</td><td>Indicates the health of this availability replica during</td><td></td><td>The values reported by this measure and their numeric equivalents are available in</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td>synchronization.</td><td></td><td>the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>NOT_ HEALTHY</td><td>0</td></tr><tr><td>PARTIALLY_ HEALTHY</td><td>1</td></tr><tr><td>HEALTHY</td><td>2</td></tr></table>
Note:
This measure reports the Measure Values listed in the table above to indicate the health status of the availability replica during synchronization. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.</td></tr>
<tr><td>Connected state:</td><td>Indicates the connection state of this availability replica with the primary/secondary availability replica.</td><td></td><td>The values reported by this measure and their numeric equivalents are available in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>DISCONNECTED</td><td>0</td></tr><tr><td>CONNECTED</td><td>1</td></tr></table>
Note:
This measure reports the Measure Values listed in the table above to indicate the connection state between the primary and secondary replicas. However, in the graph, this measure is indicated using the Numeric Values listed in the above table.
The Detailed diagnosis of this measure if enabled, lists the ReplicaID, IsLocal, Role, Last connect error number, Last</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			connect error description, and the Last connect error time.

3.1.14 Tests Disabled by Default

Tests related to database replication are disabled by default for the Microsoft SQL server.

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. Replication is typically performed to improve scalability and high availability of the database and for data warehousing and reporting purposes.

Transactional replication is the mechanism that Microsoft® SQL Server® provides to publish incremental data and schema changes to subscribers. The changes are published (the replication stream) in the order in which they occur, and typically there is low latency between the time the change is made on the Publisher and the time the change takes effect on the Subscriber. This enables a number of scenarios, such as scaling out a query workload or propagating data from a central office to remote offices and vice-versa. This form of replication always uses a hierarchical hub and spoke topology.

The following illustration is an overview of the components involved in transactional replication.

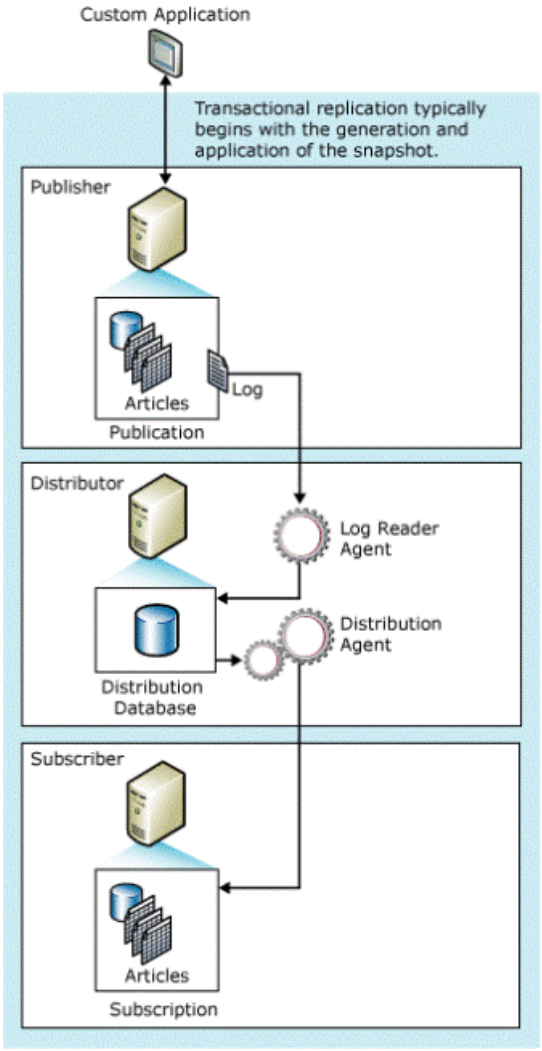


Figure 3.13: Architecture of Transactional replication

As is evident from Figure 3.13, a minimum of three server roles are required for transactional replication:

- **Publisher**: A database server that makes data available for replication (source server) is referred to as the *publisher*; a collection of one or more database objects that are enabled for replication is called a publication.
- **Distributor**: Replication is managed by the system database, which by default is called distribution. A distribution database - which can reside on the publisher, subscriber, or on a separate server - is created when you configure replication. The server that hosts the distribution database is referred to as the *distribution server or distributor*.
- **Subscriber**: One or more servers that get data and/or transactions from the publisher are called subscribers.

Depending on the complexity of the replication topology, there may be multiple Subscriber servers. Furthermore, the roles of the various replication servers can be played by one server or by individual servers

(the more common case), and it is possible for a server to play any combination of roles. Regardless, the various servers and databases must be protected to ensure that the replication stream is highly available.

Transactional replication relies on various agents to perform the tasks associated with tracking changes and distributing data. Soon after the Snapshot agent records information about the synchronization in the Distribution database, the Log Reader agent moves transactions marked for replication from the Publisher to the Distributor. These transactions are then moved to the Subscriber by the Distribution agent. If any changes are made on the Subscriber, then the Queue Reader agent moves these changes back to the Publisher.

Since replication saves the day by simplifying data recovery in the event of a database failure, care should be taken to ensure that the data on the Publisher and the Subscriber are always in sync. If one/more of the agents involved experience delays while discharging their duties, then the source and destination databases may remain out-of-sync for prolonged periods. At this juncture, if the source database becomes unavailable for any reason, the destination database cannot be used owing to the data non-sync, thereby beating the core purpose of replication! The eG Enterprise system introduces administrators to such slowdowns, much before users start complaining. Using the replication tests provided by the eG SQL Monitor, administrators can closely observe the operations of every type of agent discussed above, and can proactively capture potential latencies in their operations.

As stated earlier, these replication tests are disabled by default. To enable them, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose a 'replication test' from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

The sections that follow will discuss each test in detail.

3.1.15 Log Reader Replication Agent Test

The Replication Log Reader Agent is an executable that monitors the transaction log of each database configured for transactional replication and copies the transactions marked for replication from the transaction log into the distribution database. Each database published using transactional replication has its own Log Reader Agent that runs on the Distributor and connects to the Publisher.

If the replication process is found to take too long to complete, then, you can monitor the operations of the Log Reader agent to figure out whether/not this agent is contributing to the slowdown. This can be achieved with the help of the **Log Reader Application Agent** test. This test periodically monitors the activities of the Log Reader agent, and reveals how quickly the agent copies transactions to the Distributor; in the process, the test turns your attention to latencies (if any) in the agent's operations.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed

2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "**INSTANCE**" parameter.
5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Commands Delivered:	Indicates the rate at which the Log Reader agent delivered commands to the Distributor.	Commands/Sec	
Transactions Delivered:	Indicates the rate at which the Log Reader agent delivered transactions to the Distributor.	Trans/Sec	A high value is desired for this measure. If the value of this measure dips consistently over time, it is indicative of a slowdown.
Latency:	Indicates the current amount of time, in milliseconds, elapsed from when transactions are applied at the Publisher to when they are delivered to the Distributor.	MilliSec	Ideally, the value of this measure should be 0 or very low. A high value indicates that the Log Reader agent is taking too much time to copy transactions to the Distributor. The possible causes for this are as follows: • A large number of replicated transactions in the transaction log; • A large number of non-replication transactions; • A large number of virtual log files (VLFs) • Slow disk subsystem • Unexpected network I/O problems

Measurement	Description	Measurement Unit	Interpretation
			• Blocking by another SQL Replication Agent such as a the Distribution Cleanup Agent

3.1.16 Snapshot Replication Agent Test

The Snapshot Agent run at the Distributor and is typically used with all types of replication. It prepares schema and initial data files of published tables and other objects, stores the snapshot files, and records information about synchronization in the distribution database.

The first step towards troubleshooting a delay in database replication is to figure out at which step of the replication process the delay occurred, and which agent performed that step. Using the **Snapshot Replication Agent** test, you can determine whether/not the Snapshot agent is contributing to a slowdown (if any) in the replication process. This test, at pre-configured intervals, monitors how quickly the agent delivers transactions to the distribution database. Latencies in creation of snapshot files or in recording synchronization details in the Distributor are thus revealed.

This test is disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose a 'replication test' from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Microsoft SQL server being monitored

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "**INSTANCE**" parameter.
5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Commands Delivered:	Indicates the rate at which the Snapshot agent delivered commands to the Distributor.	Commands/Sec	
Transactions Delivered:	Indicates the rate at which the Snapshot agent delivered transactions to the Distributor.	Trans/Sec	A high value is desired for this measure. If the value of this measure dips consistently over time, it is indicative of a slowdown. A contention for memory / CPU resources on the SQL server, or high disk I/O on the server could cause snapshotting to slow down.

3.1.17 Distribution Replication Agent Test

The Distribution Agent is used with snapshot replication and transactional replication. It applies the initial snapshot to the Subscriber and moves transactions held in the distribution database to Subscribers. The Distribution Agent runs at either the Distributor for push subscriptions or at the Subscriber for pull subscriptions.

If replication is taking longer than usual, you may want to check how each of the agents involved in the replication process is performing, so that you can isolate the agent that could be causing the delay. The **Distribution Replication Agent** test helps you run frequent health checks on the Distribution agent and reveals whether latencies (if any) in the operations of this agent are the root-cause for the replication slowdown.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. **ISPASSIVE** – If the value chosen is **Yes**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Commands Delivered:	Indicates the rate at which the Distribution agent delivered commands to the Subscriber.	Commands/Sec	
Transactions Delivered:	Indicates the rate at which the Distribution agent delivered transactions to the Subscriber.	Trans/Sec	A high value is desired for this measure. If the value of this measure dips consistently over time, it is indicative of a slowdown.
Latency:	Indicates the current amount of time, in milliseconds, elapsed from when transactions are delivered to the Distributor to when they are applied at the Subscriber.	MilliSeconds	Ideally, the value of this measure should be 0 or very low. A high value indicates that the Distribution agent is taking too much time to push transactions to the Subscriber. The possible causes for this are as follows: • A series of transactions could be trying to move a large batch of commands to the Subscribers • Blocking of the Distribution Agent by another Replication Agent

Measurement	Description	Measurement Unit	Interpretation
			• Long execution times in the INS/UPD/DEL stored procedures used to apply transactions to the subscriber • SQL statements not being replicated as 'parameters'

3.1.18 Merge Replication Test

The Merge Agent is used with merge replication. It applies the initial snapshot to the Subscriber and moves and reconciles incremental data changes that occur. Each merge subscription has its own Merge Agent that connects to both the Publisher and the Subscriber and updates both. The Merge Agent runs at either the Distributor for push subscriptions or the Subscriber for pull subscriptions. By default, the Merge Agent uploads changes from the Subscriber to the Publisher and then downloads changes from the Publisher to the Subscriber.

The speed with which the Merge agent uploads/downloads changes between the Publisher and Subscriber and the count of conflicts it detects in the process influence how efficient the Merge agent is and how quickly data replication occurs. If data replication stalls, you may want to check the performance of the Merge agent to figure out possible reasons for the slowdown. The **Merge Replication** test enables this analysis. This test monitors the operations of the Merge agent and reports how quickly changes were uploaded and downloaded by the agent and how many conflicts were detected (per second) while merging. These statistics are useful when trying to determine whether/not issues with the Merge agent are causing problems in replication.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Downloaded changes:	Indicates the number of rows per second replicated from the Publisher to the Subscriber.	Rows/Sec	A high value is desired for these measures. If the value of these measures dips consistently over time, it could indicate a slowdown.
Uploaded changes:	Indicates the number of rows per second replicated from the Subscriber to the Publisher.	Rows/Sec	
Conflicts:	Indicates the number of conflicts per second occurring during the merge process. .	Conflicts/Sec	Merge replication allows multiple nodes to make autonomous data changes, so situations exist in which a change made at one node may conflict with a change made to the same data at another node. In other situations, the Merge Agent encounters an error such as a constraint violation and cannot propagate a change made at a particular node to another node. Ideally, the rate of such conflicts should be low. A high value or a steady increase in this value indicates that too many conflicts have been detected, but only a few have been resolved. A large number of merge conflicts and delays in their resolution can adversely impact replication. The Merge Agent detects conflicts by using the lineage column of the MSmerge_ contents system table; if column-level tracking is enabled for an article, the COLV1 column is also used. These columns contain metadata about when a row or column is inserted or updated, and about which nodes in a merge replication topology made

Measurement	Description	Measurement Unit	Interpretation
			changes to the row or column. As the Merge Agent enumerates changes to be applied during synchronization, it compares the metadata for each row at the Publisher and Subscriber. The Merge Agent uses this metadata to determine if a row or column has changed at more than one node in the topology, which indicates a potential conflict. After a conflict is detected, the Merge Agent launches the conflict resolver specified for the article with a conflict and uses the resolver to determine the conflict winner. The winning row is applied at the Publisher and Subscriber, and the data from the losing row is written to a conflict table. Conflicts are resolved automatically and immediately by the Merge Agent unless you have chosen interactive conflict resolution for the article.

3.1.19 Replication Agents Test

This test reports the number of replication agents that are currently running on a target Microsoft SQL server.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of

the "instance" parameter.

5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Running Agents:	Indicates the number of replication agents currently running.	Number	

3.2 The MS SQL Memory Structures Layer

This layer tracks the health of the memory and buffer structures of an MS SQL server tests shown in Figure 3.14. The details of the tests are available in the following sections.

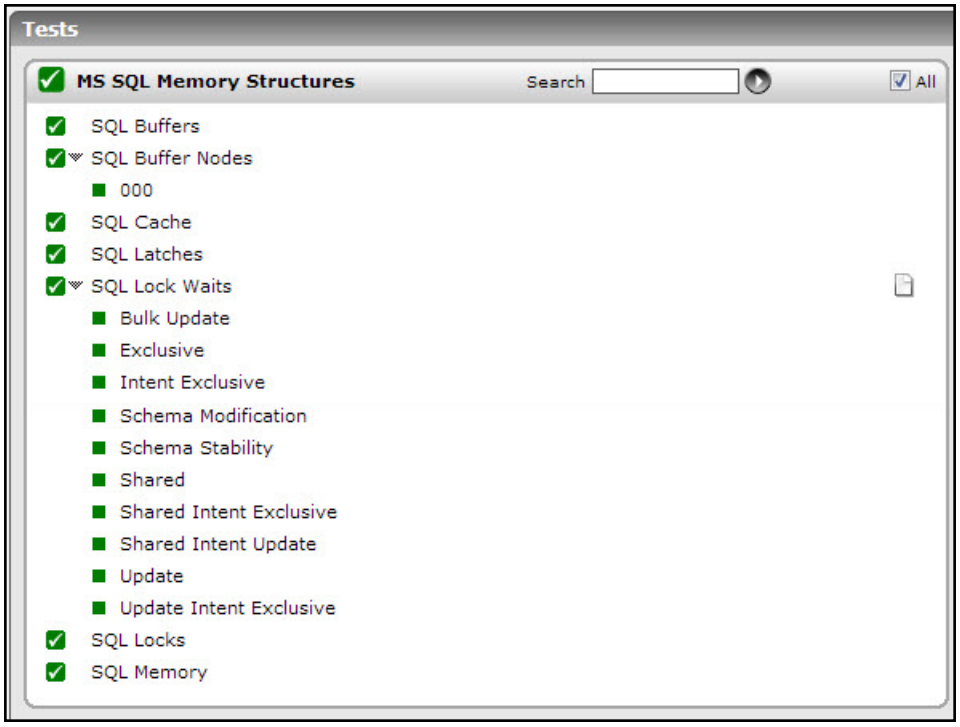


Figure 3.14: Tests pertaining to the MS SQL Memory Structures layer

3.2.1 SQL Memory Test

This test monitors the memory usage of an Microsoft SQL server.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total server memory:	This value indicates the total amount of memory that is being currently used by the server.	MB	An unusually large usage of memory by the server is a cause of concern. Further analysis is required to determine if specific users or queries are consuming excess memory.
Target server memory:	This value indicates the total amount of dynamic memory, which the server can consume.	MB	If, over time, the Total server memory measure is less than the Target server memory counter, then this means that SQL server has enough memory to run efficiently. On the other hand, if the Total server memory measure is greater than the Target server memory counter,

Measurement	Description	Measurement Unit	Interpretation
			this indicates that SQL Server may be under memory pressure and could use access to more physical memory.
Sql cache memory:	This value indicates the total amount of dynamic memory that the server is using for the dynamic SQL cache.	MB	The amount of data cache available to SQL Server can significantly affect SQL Server's performance. If the dynamic SQL cache memory usage is low, consider tuning the cache management parameters of SQL server.
Optimizer memory:	This value indicates the total amount of dynamic memory, which the server is using for query optimization.	MB	If the optimizer memory usage is low, consider tuning the optimizer memory management parameters of the Microsoft SQL server.
Max workspace memory:	This value indicates the maximum amount of memory allocated for the execution of processes. This memory is used primarily for operations like hash, sort and create index.	MB	This parameter is useful in conjunction with the grant workspace memory. When the grant workspace memory reaches the max workspace memory then we should consider tuning this.
Lock memory:	This value indicates the total amount of dynamic memory, which the server has allocated for locks.	MB	If the memory allocated for locks is less and there is a contention/wait for a lock, try tuning the lock memory management parameters of the Microsoft SQL server
Grant workspace memory:	This value indicates the total amount of memory granted for the execution of processes. This memory is used for hash, sort and create index operations.	MB	If the grant workspace memory is nearing the maximum workspace memory then the maximum workspace memory may have to be increased.
Connection memory:	This value indicates the total amount of dynamic	MB	If the memory allocated for connection is less, try tuning the memory

Measurement	Description	Measurement Unit	Interpretation
	memory, which the server is using for maintaining connections.		management parameters of the Microsoft SQL server.
Memory grants pending:	Indicates the total number of processes waiting for a workspace memory grant.	Number	In general, if you have any processes queuing waiting for memory, you should expect degraded performance. The ideal situation for a healthy server is no outstanding memory grants - i.e., the value of this measure should ideally be 0.

3.2.2 SQL Locks Test

This test monitors the locking activity of various transactions supported by an Microsoft SQL server.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
6. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable" by the agent if the server is not up.

7. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

   The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

   - The eG manager license should allow the detailed diagnosis capability
   - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Avg wait time for locks:	This gives the average amount of wait time for each lock request that resulted in a wait.	Secs	A high value may indicate that there is contention for locks in the system. When the average wait time for locks is high, users may have to wait for their transactions to complete. This metric reports the average wait time for a variety of locks, including database, extent, key, page, RID, and table. When the average wait time is high, use the SQL profiler to identify which lock(s) is causing transaction delays. The detailed diagnosis of the Avg wait time for locks measure, if enabled, provides the average wait time for each lock type.
Lock requests:	This value gives the number of new locks and lock conversions requested from the lock manager per second.	Reqs/Sec	A high value indicates that there is high locking activity in the system and may need close scrutiny for the type of locks being requested.
Lock waits:	This value indicates the number of lock requests per	Waits/Sec	A high value of waits can have an adverse impact on application performance. Possible reasons for this behavior could be:

Measurement	Description	Measurement Unit	Interpretation
	second that could not be satisfied immediately and required the caller to wait before being granted the lock.		• inadequate number of locks available in the database, • unusually high locking behavior of applications accessing the database, • improper database application design, etc.
Deadlocks:	Number of lock requests/sec that resulted in a deadlock	Deadlocks/Sec	A deadlock may arise due to various situations including bad design of queries and deficient coding practices. A deadlock is a situation where both/all the lock requestors are in a mutual or a multi- way tie. Any deadlocks are detrimental to database application performance.
Lock timeouts:	indicates the number of lock requests per second that timed out, including requests for NOWAIT locks.	Timeouts/Sec	The LOCK_TIMEOUT setting allows an application to set a maximum time that a statement waits on a blocked resource. When a statement has waited longer than the LOCK_ TIMEOUT setting, the blocked statement is canceled automatically, and error message 1222 (Lock request time-out period exceeded) is returned to the application. Any transaction containing the statement, however, is not rolled back. A high value for this measure indicates that many statements were cancelled, as they could not acquire a lock on a resource for a long time. This may be due to another extent lock holding the resource (An extent lock is the one which locks multiple resources). In such a situation, use the detailed diagnosis of the SQL Blocker

Measurement	Description	Measurement Unit	Interpretation
			Processes test to know which statements have been blocked for a long time, and which statements are blocking them.

The detailed diagnosis of the *Lock requests* measure, if enabled, provides the rate of locks for each lock type (see Figure 3.15).

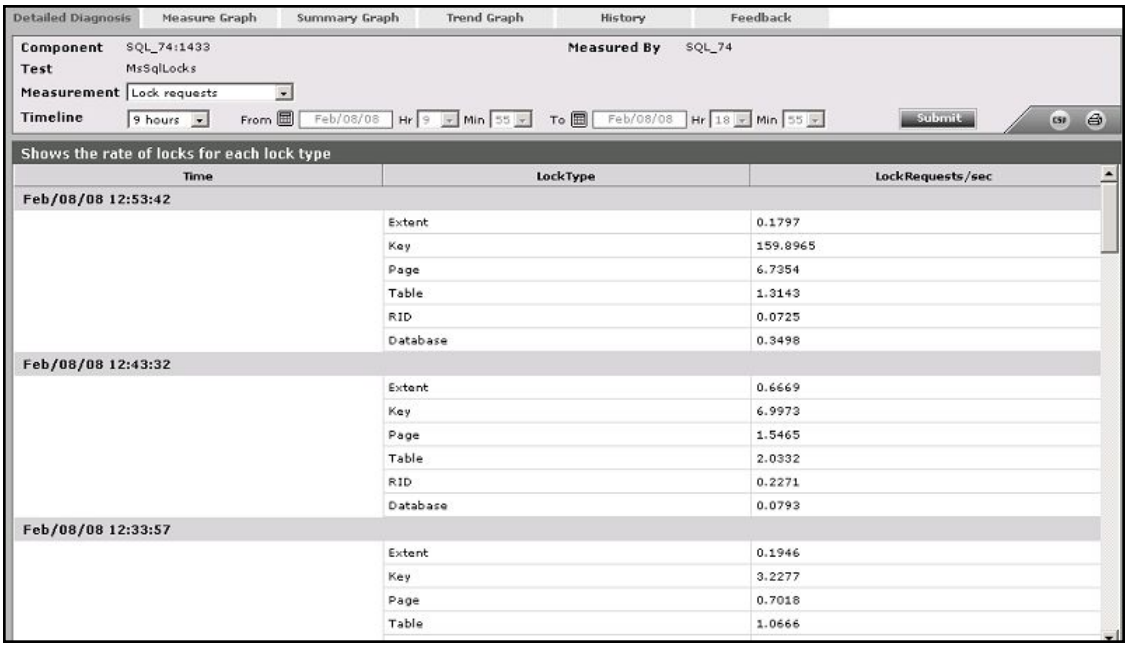


Figure 3.15: The detailed diagnosis of the Lock requests measure

The detailed diagnosis of the *Lock waits* measure, if enabled, displays the rate of lock waits for each lock type (see Figure 3.16).

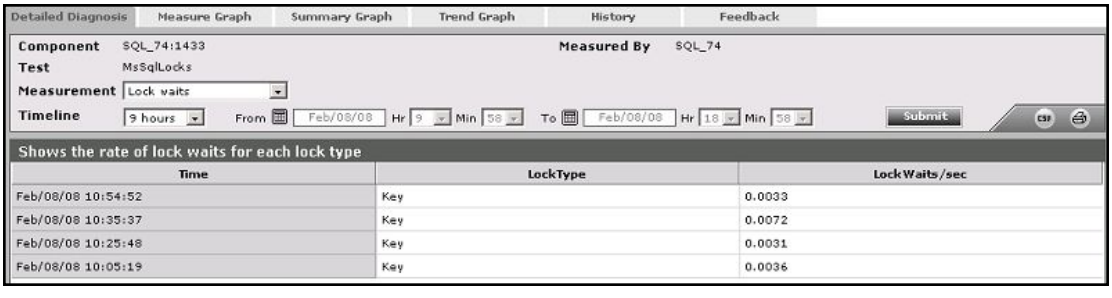


Figure 3.16: The detailed diagnosis of the Lock waits measure

3.2.3 SQL Latches Test

A latch is a “lightweight lock”. A latch acts like a lock, in that its purpose is to prevent data from changing unexpectedly. For example, when a row of data is being moved from the buffer to the SQL Server storage engine, a latch is used by SQL Server during this move to prevent the data in the row from being changed during this very short time period. This not only applies to rows of data, but to index information as well, as it is retrieved by SQL Server.

Just like a lock, a latch can prevent SQL Server from accessing rows in a database, which can hurt performance. Because of this, latch wait time must be minimized.

The **SQL Latches** test makes the following measures of latch activity for an Microsoft SQL database.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.
6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Latch wait rate:	This is the number of latch requests per second that could not be granted immediately.	Waits/Sec	The latch activity may vary from one server to another. Hence, it is important to get baseline numbers for this metric, so that you can compare "typical" latch activity against what is happening currently. If latch activity is higher than expected, this often indicates one of two problems. First, it may mean that the SQL Server could use more memory. If latch activity is high, check the buffer cache hit ratio is. If it is below 99%, the SQL server could probably benefit from more memory. If the hit ratio is above 99%, then it could be the I/O system that is contributing to the problem, and a faster I/O system might improve server's performance.
Total latch wait time:	This metric denotes the total wait time for latch requests that had to wait in the last second.	Secs	Ideally, this value should be close to 0. The larger this value is, the more contention there is for latches and worse the performance of the database. If the wait time is high, check the *SYSPROCESSES* table of the database to see which latches are seeing most contention.

3.2.4 SQL Cache Test

This internal test monitors the usage of buffer memory of an MS SQL server.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed

2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "instance" parameter.
5. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
7. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

   The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

   - The eG manager license should allow the detailed diagnosis capability
   - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Cache hit ratio:	This value indicates the percentage of pages found in the cache without having to read from disk.	Percent	The ratio is the total number of cache hits divided by the total number of cache lookups since SQL Server was started. Because reading from the cache is less expensive than reading from disk, you want the ratio to be high. The higher this value is, the better. Generally, you can increase the cache hit ratio by increasing the amount of memory

Measurement	Description	Measurement Unit	Interpretation
			available to the SQL Server.
Objects in cache:	This value indicates the number of objects found in the cache currently.	Number	The higher the count, the better the performance of the server, as it does not need to read the requested object from disk.
Log cache hit ratio:	This value indicates the percentage of log cache reads satisfied from the log cache.	Percent	A good cache hit ratio indicates that the log cache is performing well and a low value indicates that the server needs to be tuned.

The detailed diagnosis of the *Cache hit ratio* measure, if enabled, provides the cache hit ratio for each cache type (see Figure 3.17). A high cache hit ratio is an indicator of the SQL server's good health. The information provided by Figure 3.17 will therefore reveal the cache types that were effective and those that were not.

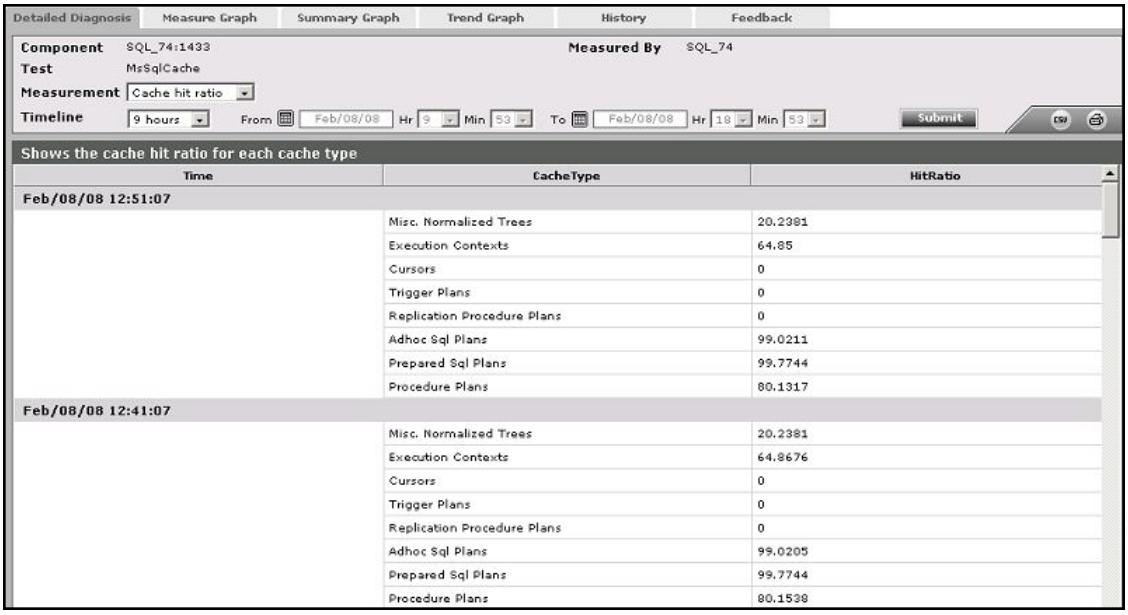


Figure 3.17: The detailed diagnosis of the Cache hit ratio measure

The detailed diagnosis of the *Objects in cache* measure , if enabled, provides the number of objects available in each cache type (see Figure 3.18). This again is an indicator of the effectiveness of the cache types.

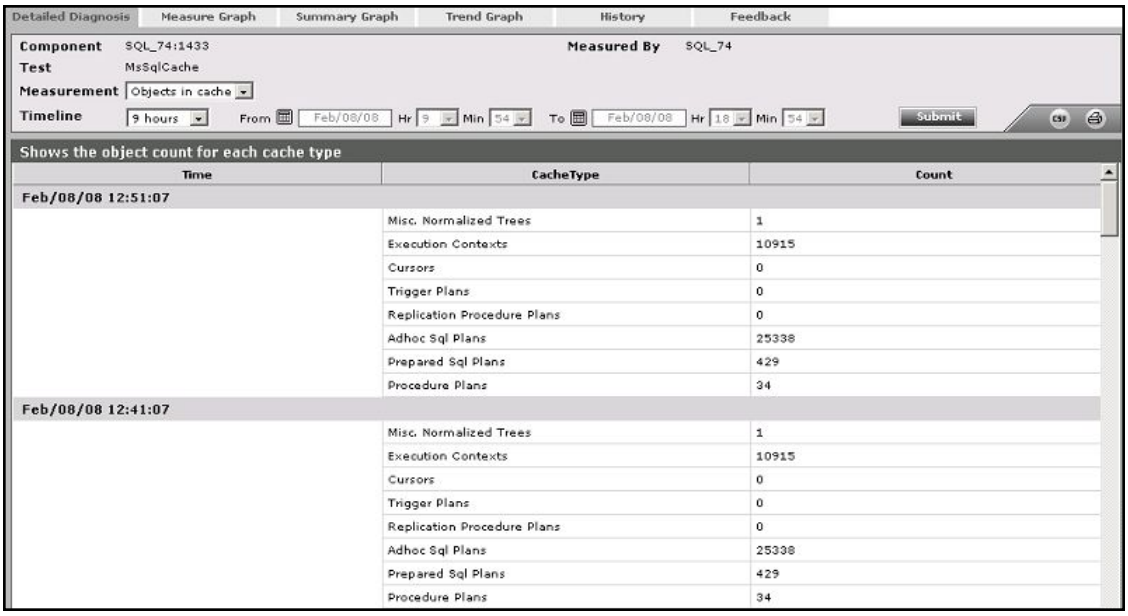


Figure 3.18: The detailed diagnosis of the Objects in cache measure

3.2.5 SQL Buffers Test

This internal test also monitors the usage of buffer memory of an Microsoft SQL server.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target Microsoft SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled Microsoft SQL server, set the **SSL** flag to **Yes**.
5. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the “instance” parameter.
6. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.

7. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Buffer cache hit ratio:	This value indicates the percentage of pages that are served from the server's buffer cache (i.e., without requiring a read from the disk).	Percent	This value is the ratio of the total number of cache hits to the total number of cache lookups since the server was started. Because reading from the cache is much less expensive than reading from disk, this ratio should ideally be high. Generally, one can increase the buffer cache hit ratio by increasing the amount of memory available to SQL Server.
Page reads:	This indicates the number of database pages that were physically read per second.	Pages/Seconds	Because physical I/O is expensive, one may be able to minimize the cost, either by using a larger data cache, intelligent indexes, more efficient queries, or by changing the database design.
Page writes:	This indicates the number of database page writes that are issued per second.	Writes/Seconds	Page writes are generally expensive. Reducing page- write activity is important for optimal tuning. One way to do this is to ensure that you do not run out of free buffers in the free buffer pool. If you do, page writes will occur while waiting for an unused cache buffer to flush.
Target Pages:	The ideal number of pages that should be in the buffer pool for optimal performance	Number	

Measurement	Description	Measurement Unit	Interpretation
Total pages:	The current total number of pages in the buffer pool. This value includes the number of pages in the database pool, free pool, and stolen list.	Number	For optimum performance, the total pages must be close to the target pages value. **This measure will not be available for Microsoft SQL Server 2012.**
Free pages:	The total number of pages currently in all free lists	Number	**This measure will not be available for Microsoft SQL Server 2012.**
Stolen pages:	The number of buffer pool pages used to satisfy other server memory requests	Number	**This measure will not be available for Microsoft SQL Server 2012.**
Lazy writes rate:	Indicates the number of lazy writes per second.	Writes/Seconds	Lazy writes are buffers written by the lazy writer. The lazy writer is a system process that flushes out batches of dirty, aged buffers (buffers that contain changes that must be written back to disk before the buffer can be reused for a different page) and makes them available to user processes. The lazy writer eliminates the need to perform frequent checkpoints in order to create available buffers. Keeping the number of lazy writes low can enhance performance. A supply of buffers available for immediate use keeps the number of lazy writes low. Before a requested page can be brought into memory, a free buffer must be available in the buffer pool. If no free buffers are available, an existing buffer must be reused. When an existing buffer has to be reused, many buffer pages might have to be searched in

Measurement	Description	Measurement Unit	Interpretation
			order to locate a buffer to reclaim for use. If the buffer found is marked as dirty or modified, the buffer manager must first write the changes to disk before the page can be reused and assigned to the requesting process. This results in a wait for the requesting process. Waiting processes can degrade performance.
Page life expectancy:	Indicates the number of seconds a page will stay in the buffer pool without references.	Secs	A high value of this measure indicates that your page stays longer in the buffer pool (area of the memory cache), thereby leading to higher performance. This is because, every time a request comes in, the probability of that request been served by the cache is higher; this in turn significantly reduces direct disk accesses, which are relatively more expensive operations. A value too low for your system indicates that pages are being flushed from the buffer pool too quickly. The target on an OLTP system should be at least 300 seconds (5min). When under 300 seconds, this may indicate poor index design (leading to increased disk I/O and less effective use of memory) or, simply, a potential shortage of memory.
Free list stalls:	Indicates the number of requests per second that had to wait for a free page.	Stalls/Sec	Free list stall rates of 3 or 4 per second indicate too little SQL memory available.

3.2.6 SQL Buffer Nodes Test

This test is specific to Microsoft SQL Server 2005 (or above).

Microsoft SQL Server 2005 (or above) is non-uniform memory access (NUMA) aware. As clock speed and the number of processors increase, it becomes increasingly difficult to reduce the memory latency required to use this additional processing power. NUMA architecture provides a scalable solution to this problem. SQL Server 2005 (or above) has been designed to take advantage of NUMA-based computers without requiring any application changes. This database server groups schedulers to map to the grouping of CPUs, based on the hardware NUMA boundary exposed by Windows. For example, a 16-way box may have 4 NUMA nodes, each node having 4 CPUs. This allows for a greater memory locality for that group of schedulers when tasks are processed on the node. With SQL Server 2005 (or above) you can further subdivide CPUs associated with a hardware NUMA node into multiple CPU nodes. This is known as soft-NUMA. There is one SQL Server memory node for each physical NUMA node. When a thread running on a specific hardware NUMA node allocates memory, the memory manager of SQL Server tries to allocate memory from the memory associated with the NUMA node for locality of reference. Similarly, buffer pool pages are distributed across hardware NUMA nodes. It is more efficient for a thread to access memory from a buffer page that is allocated on the local memory than to access it from foreign memory.

To understand the local vs. foreign memory distribution in SQL Server better, assume that the computer has 16 gigabytes (GB) of memory. Other applications including Windows have consumed some of the memory from each node, SQL Server has assigned some memory for its processes outside of the buffer pool, and SQL Server has 10 GB of memory to assign to the buffer pool. The buffer pool memory is divided among four physical NUMA nodes, N0, N1, N2, and N3, each with the following local memory available:

- N0 – 1 GB
- N1 – 3 GB
- N2 – 3 GB
- N3 – 3 GB

In the above configuration, all nodes will eventually allocate and use 2.5 GB of memory; however, node N0 will end up with 1.0 GB of its own memory, and 1.5 GB of memory from other nodes.

The SQL Buffer Nodes test reports information about buffer pool page distribution for each NUMA node on MSSQL Server 2005 (or above).

Target of the test : A Microsoft SQL server 2005 (or above)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every NUMA node on the Microsoft SQL Server 2005 (or above)

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of

the "instance" parameter.

5. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target Microsoft SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled Microsoft SQL server, set the **SSL** flag to **Yes**.
6. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
7. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Database pages:	Indicates the number of pages that are currently in the buffer pool with database content.	Number	
Foreign pages:	Indicates the number of pages that were currently received from other NUMA nodes.	Number	If the SQL server is running on non-NUMA hardware, then the value of this measure will be 0. Foreign pages will not be used during ramp-up because they can frequently be transferred to the owning node and become local to that node. When the value of max server memory is reached, some nodes may have foreign memory, but once the memory target is achieved, the buffer pool will treat local and foreign memory identically. For example, under memory pressure, the buffer pool will not make any effort to free up foreign memory pages before local memory pages. If the value of this measure increases

Measurement	Description	Measurement Unit	Interpretation
			consistently, it indicates that local memory on the node is inadequate.
Free pages:	Indicates the total number of pages that are currently free on this NUMA node.	Number	If the value of this measure dips consistently, it is indicative of insufficient memory on the node. You might want to consider resizing the buffer pool in this case.
Page life expectancy:	Indicates the number of seconds a page currently stayed in the buffer pool without references.	Secs	
Stolen pages:	Indicates the number of pages that were currently used for miscellaneous server purposes (stolen from the buffer pool) on this node.	Number	
Target pages:	Indicates the ideal number of pages in the buffer pool on this node.	Number	
Total pages:	Indicates the total number of committed pages that currently exist in the buffer pool on this node.	Number	

3.2.7 SQL Lock Waits Test

lock wait event occurs when a user requests for a resource that is already locked by another user, forcing the former to wait until the latter releases the lock. Lock wait events on a database need to be minimal. If a lock is held on a resource for too long a time, all other requests will be denied access to that resource, thereby causing critical operations to fail. Moreover, if the number of lock waits grows over time, it will consequently increase the length of the pending requests queue; a long request queue may not only cause the unnecessary erosion of valuable server resources, it may also choke the database server, thereby significantly impacting

the quality of the user experience with the server. It is therefore imperative that the lock wait events are monitored, and issues related to such events immediately brought to the attention of administrators.

This test monitors the lock wait events for each lock type, and promptly alerts administrators to a sudden/steady increase in the number and duration of such events. The lock types which will form the descriptors of this test are discussed below:

Descriptor	Lock Type	Description
S	Shared locks	Shared locks are held on data being read under the pessimistic concurrency model. While a shared lock is being held other transactions can read but cannot modify locked data. After the locked data has been read the shared lock is released, unless the transaction is being run with the locking hint (READCOMMITTED, READCOMMITTEDLOCK) or under the isolation level equal or more restrictive than Repeatable Read.
U	Update locks	Update locks are a mix of shared and exclusive locks. When a DML statement is executed SQL Server has to find the data it wants to modify first, so to avoid lock conversion deadlocks an update lock is used. Only one update lock can be held on the data at one time, similar to an exclusive lock. But the difference here is that the update lock itself can't modify the underlying data. It has to be converted to an exclusive lock before the modification takes place.
X	Exclusive locks	Exclusive locks are used to lock data being modified by one transaction thus preventing modifications by other concurrent transactions. You can read data held by exclusive lock only by specifying a NOLOCK hint or using a read uncommitted isolation level. Because DML statements first need to read the data they want to modify you'll always find Exclusive locks accompanied by shared locks on that same data.
I	Intent locks	Intent locks are a means in which a transaction notifies other transaction that it

Descriptor	Lock Type	Description
		is intending to lock the data. Thus the name. Their purpose is to assure proper data modification by preventing other transactions to acquire a lock on the object higher in lock hierarchy. What this means is that before you obtain a lock on the page or the row level an intent lock is set on the table. This prevents other transactions from putting exclusive locks on the table that would try to cancel the row/page lock.
Sch	Schema locks	There are two types of schema locks: • Schema stability lock (Sch-S): Used while generating execution plans. These locks don't block access to the object data. • Schema modification lock (Sch-M): Used while executing a DDL statement. Blocks access to the object data since its structure is being changed.
SIX	Shared with Intent Exclusive	A transaction that holds a Shared lock also has some pages/rows locked with an Exclusive lock
SIU	Shared with Intent Update	A transaction that holds a Shared lock also has some pages/rows locked with an Update lock
UIX	Update with Intent Exclusive	A transaction that holds an Update lock also has some pages/rows locked with an Exclusive lock

Note:

This test is applicable only to Microsoft SQL Server 2005 (and above).

Target of the test : A Microsoft SQL server 2005 (and above)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for lock wait type on the Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.

3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER.**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.
    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:
    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Current lock waits:	Indicates the current number of lock wait events for this lock type.	Number	Ideally, this value should be very low. A high value or a consistent increase in the value may choke the database server and severely hamper its overall performance. Therefore, if the value of this measure is high, you might first need to identify what is causing the lock waits. For this purpose, you can use the detailed diagnosis of this measure. The detailed diagnosis leads you to the exact object the lock events are waiting on, the user who holds a lock on that object, and the query that initiated the lock. This way, inefficient queries can be identified and fine- tuned. Given below are some tips for minimizing lock waits: • Keep all Transact-SQL transactions as short as possible. • To reduce the amount of time that tables are locked, which hurts concurrency and performance, avoid interleaving reads and database changes within the same transaction. Instead, try to do all your reads first, then perform all of the database changes (*UPDATES, INSERTS, DELETES*) near the end of the transaction. This helps to minimize the amount of time that exclusive locks are held. • Use clustered indexes on heavily used tables. • Make appropriate use of non-clustered indexes • Try to avoid Transact- SQL statements that affect large numbers

Measurement	Description	Measurement Unit	Interpretation
			of rows at once, especially the *INSERT* and *UPDATE* statements. • Try to have your *UPDATE* and *DELETE* statements use an index. • When using nested transactions, avoidcommit and rollback conflicts.
Avg. wait time for locks:	Indicates the duration of the lock wait events for this lock type.	Milliseconds	Ideally, this value should be very low. A high value or a consistent increase in the value may choke the database server and severely hamper its overall performance. Therefore, if the value of this measure is high, you might first need to identify what is causing the lock waits. For this purpose, you can use the detailed diagnosis of the *Current lock waits* measure. The detailed diagnosis leads you to the exact object the lock events are waiting on, the user who holds a lock on that object, and the query that initiated the lock. This way, inefficient queries can be identified and fine-tuned.

3.3 The MS SQL Workload Layer

The tests mapped to this layer help analyze the query, transaction, and process workload on the SQL server.

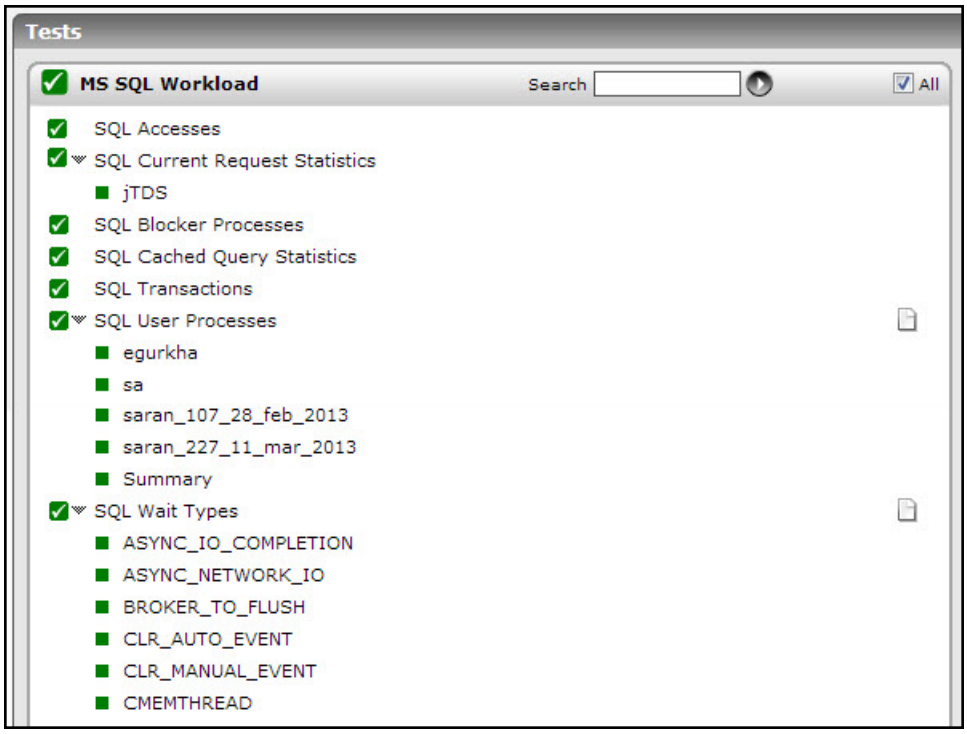


Figure 3.19: The tests mapped to the MS SQL Workload layer

3.3.1 SQL Accesses Test

This test monitors various critical metrics regarding accesses to the MS SQL database.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the “instance” parameter.
5. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target MS SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled MS SQL server, set the **SSL** flag to **Yes**.
6. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a

passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

7. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Full table scans:	This metric represents the number of unrestricted full scans being handled by the server. The scans can either be base table or full index scans and the value represents the sum of all scans for all the database instances being handled by the server.	Scans/Sec	Generally it is better to have fewer table scans for a database. In many cases, the SQL server itself will perform a few full table scans on a regular basis for internal use. To detect anomalies, check for random full table scans that may represent the behavior of applications using the database server. If an unusually high number of full table scans are noticed, use the Profiler and Index Tuning Wizard to determine what is causing these scans, and if adding any indexes can help reduce the table scans.
Page splits:	This value represents the rate of page splits occurring in a database server as the result of index pages overflowing.	Splits/Sec	If the rate of page splits is high, consider increasing the fill factor of the indexes.
Sql compilations:	This value is the rate of SQL compilations happening in the database.	Reqs/Sec	SQL compilations are a normal part of an SQL Server's operation. But because compilations take up CPU and other resources, the SQL server attempts to reuse as many execution plans in cache as possible (execution plans are created when compilations occur). The more that execution plans

Measurement	Description	Measurement Unit	Interpretation
			are reused, the less overhead there is on the server, and the faster overall performance is. A high number of SQL compilations indicates that the server is very busy. Compilations can be forced if object schema changes, if previously parallelized execution plans have to run serially, if statistics are recomputed, or if a number of other things occur. In some cases, it is possible to reduce the number of unnecessary compilations.
Sql recompilations:	This value is the rate of SQL recompilations happening in the database server.	Reqs/Sec	The lower the rate of SQL recompilations, the better the database server performance can be.
Batch requests:	This value is the rate of batch requests being handled by the database server.	Reqs/Sec	This metric is a good indicator of how busy an SQL Server is. This metric is generally in step with the server's CPU usage. If the number of batch requests processed is very high (1000 requests/sec or higher), check for CPU and network bottlenecks that can be caused by the high activity rate. The capacity of an MS SQL database server to handle batch requests will depend on the hardware capabilities (CPU, memory, network interface speeds, etc.).
Work files rate:	Indicates the number of workfiles that are created per second.	Workfiles/Sec	Workfiles are used for temp record storage during hash operations. They are created in memory but can overflow or spill to disk (to tempdb, not to separate OS- level files). A value greater than 20 for this measure indicates trash in the tempdb or poorly coded queries.

Measurement	Description	Measurement Unit	Interpretation
Work tables rate:	Indicates the number of worktables created per second.	Worktables/Sec	The relational engine may need to build a worktable to perform a logical operation specified in an SQL statement. Worktables are typically generated for certain GROUP BY, ORDER BY, or UNION queries. For example, if an ORDER BY clause references columns not covered by any indexes, the relational engine may need to generate a worktable to sort the result set into the order requested. Worktables are built in tempdb and are dropped automatically at the end of the statement.
Cache requests for work table creation:	Indicates the percentage of work tables created where the initial two pages of the work table were not allocate but were immediately available from the work table cache.	Percent	When a query execution plan is cached, the tempdb work tables required by the plan, if any, are often cached. When a work table is cached, the table is truncated (from the previous execution of the code) and up to nine pages remain in the cache for reuse. This improves the performance of the next execution of the query. A value less than 90% for this measure may indicate insufficient memory; this is because, when memory is low, the database server engine drops execution plans and their corresponding work tables from the cache. This may have caused the low cache hit ratio for work tables. On 32-bit systems, a low value for this measure may also hint at the need for upgrading to 64-bit.
	Indicates the percentage of scans that were initiated to search for free space in which to insert a new record	Percent	This measure represents inserts into a table with no physical ordering of the rows. A table with no ordering, without

Measurement	Description	Measurement Unit	Interpretation
	fragment.		a clustered index, is known as a heap table. Inserts into heaps will require SQL Server to perform freespace scans to identify pages with free space to insert rows. A heap table also requires an additional, internal column called an uniquifier to be generated for each row inserted. Extra processing is required to define and store a heap table since SQL Server normally uses the clustered index as a storage mechanism for the table data. Freespace scans have an additional I/O expense for inserts and can possibly cause contention on the GAM, SGAM, and PFS pages when there are many connections inserting. It is usually recommended that you physically order the table rows by using a clustered index on the table. A low value is hence desired for this measure. If the value of this measure is very high, you have to quickly investigate the reasons for the same and rapidly figure out what needs to be done to control the free space scans. Towards this end, you first need to identify the I/O-intensive queries executing on the SQL database. For this, you can use the detailed diagnosis of the Avg physical reads measure of the SQL Cached Queries test. The detailed diagnosis of this test reveals which queries are serviced by direct database accesses (and not by the cache). Since such queries are bound to increase processing overheads, optimizing these queries will not only conserve

Measurement	Description	Measurement Unit	Interpretation
			resources, but will also reduce free space scans. Secondly, use the detailed diagnosis of the SQL Missing Indexes test to precisely pinpoint which columns on which tables are not indexed. Unindexed columns can cause queries to take more time and consume more I/O resources. Identifying such columns and indexing them can help reduce I/O and free space scans. Thirdly, you can use the detailed diagnosis of the SQL Unused Indexes test to know which indexes are not used effectively, and are hence increasing I/O overheads during query execution. By ensuring that such indexes are used, you can reduce I/O and free space scans.
SQL cancelled rate:	Indicates the number of SQL queries that were cancelled per second.	Cancelled/Sec	Users my cancel the currently executing query/batch request at any time. When such cancellations occur, an attention event occurs. An attention is a request by the client to end the currently running request. Typically, an attention event can occur when a cancel request, client-interrupt request, or a broken client connection has occurred. A high value for this measure may hence result in a large number of attention events. If the query or batch that was cancelled was in an explicit user transaction (BEGIN TRAN …. END TRAN), these attention events could result in open transactions and severe blocking problems.

3.3.2 SQL Blocker Processes Test

One common problem encountered with databases is blocking. Suppose that process A is modifying data that process B wants to use. Process B will be blocked until process A has completed what it is doing. This is only one type of blocking situation; others exist and are common. What matters to a database administrator is identifying when blocking is a problem and how to deal with it effectively. When blocking is bad enough, users will notice slowdowns and complain about it. With a large number of users, it is common for tens or hundreds of processes to be blocked when slowdowns are noticed. Killing these processes may or may not solve the problem because 10 processes may be blocked by process B, while process B itself is blocked by process A. Issuing 10 kill statements for the processes blocked by B probably will not help, as new processes will simply become blocked by B. Killing process B may or may not help, because then the next process that was blocked by B, which is given execution time, may get blocked by process A and become the process that is blocking the other 9 remaining processes. When you have lots of blocking that is not resolving in a reasonable amount of time you need to identify the root blocker, or the process at the top of the tree of blocked processes. Imagine again that you have 10 processes blocked by process B, and process B is blocked by process A. If A is not blocked by anything, but is itself responsible for lots of blocking (B and the 10 processes waiting on B), then A would be the root blocker. (Think of it as a traffic jam. Figure 3.20 will help) Killing A (via kill) is likely to unblock B, and once B completes, the 10 processes waiting on B are also likely to complete successfully. The SQL Blocker Processes test monitors the number of root blocker processes in a database.

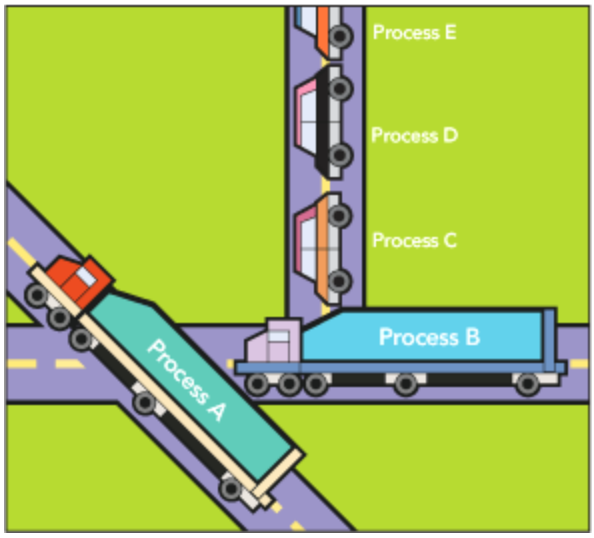


Figure 3.20: The traffic jam analogy representing blocking

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed

2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **BLOCKED SESSION COUNT** – Specify the minimum number of sessions a process should block for this test to count that process as a root blocker. For instance, if you specify 10 here, it indicates that the Number of rootblockers measure of this test will include only those processes that are blocking 10 or more sessions.
13. **MAX BLOCKING TIME SECS** – If a process is blocked for or beyond the duration (in seconds) specified here, then this test will count that process as a process that has been blocked for the maximum time. The details of such processes will then be captured and displayed as part of the detailed diagnosis of the Max waiting time measure. For example, if you specify 120 seconds here, then the detailed diagnosis of the Max waiting time measure will display the details of all processes that were blocked for 2 minutes and above.
14. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated. The default is *1:1*. This indicates that, by default, detailed measures will be generated every

time this test runs, and also every time the test detects a problem. Typically, detailed diagnosis frequencies are set globally, using the **DIAGNOSIS CONFIGURATION** page that appears when the Configure -> Diagnosis Settings menu sequence is followed. This global setting can be overridden at the test-level using the **DD FREQUENCY** parameter. To disable the detailed diagnosis capability for a test, you can set this parameter to 0:0.

15. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of root blockers:	Indicates the number of root blocker processes.	Number	Usually, the number of root blocker processes should be low. If this value increases suddenly, this is a cause for concern. Likewise, if a root- blocker process has been blocking other processes for a long time, it is a reason for further investigation. The detailed diagnosis for this test, if enabled, provides details of the root blocker processes - their SPIDs, programs running these processes, and the queries being issued by these processes. It is usually the case that killing any root-blocker process that has been running for a long while will get the database running well again.
Blocked processes:	Indicates the number of processes that are blocked by the root blockers.	Number	Use the detailed diagnosis of this measure to know which processes are blocked.

Measurement	Description	Measurement Unit	Interpretation
Max waiting time:	Indicates the waiting time – i.e., blocked time – of that process (es) that was blocked for the maximum duration.	Secs	If the value of this measure matches or exceeds the **MAX BLOCKING TIME** configuration of this test, it indicates that one/more processes have been blocked for a very long time. You can then use the detailed diagnosis of this measure to identify these blocked processes and figure out who initiated such processes and their resource usage. Processes that are resource hogs can thus be identified.

3.3.3 SQL Transactions Test

This test reports the number of transactions active in an instance of the Database Engine, and the effects of those transactions on resources such as the snapshot isolation level row version store in *tempdb*.

Transactions are logical units of work; a set of operations that must either all succeed or all be erased from a database in order to maintain the logical integrity of the data. All modifications of data in SQL Server databases are made in transactions. When a database is set to allow snapshot isolation level, SQL Server must maintain a record of the modifications made to each row in a database. Each time a row is modified, a copy of the row as it existed before the modification is recorded in a row version store in *tempdb*. Many of the measures of the MsSqlTransTest monitor the size and rate of growth of the following row version stores in *tempdb*:

- The online index build version store is used for online index builds in all databases.
- The common version store is used for all other data modification operations in all databases.

Note:

This test again, is specific to Microsoft SQL Server 2005 (or above).

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL Server 2005 (or above) that is being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.

3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
5. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
6. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target Microsoft SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled Microsoft SQL server, set the **SSL** flag to **Yes**.
7. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Free space in tempdb:	Indicates the amount of space that is currently free in tempdb.	KB	There must be enough free space in **tempdb** to hold both the snapshot isolation level version store and all new temporary objects created in this instance of the Database Engine. When the value of this measure decreases, the Database Engine forces the version stores to shrink. During the shrink process, the longest running transactions that have not yet generated row versions are marked as victims. A message 3967 is generated in the error log for each victim transaction. If a transaction is marked as a victim, it can no longer read the row versions in the version store. When it attempts to read row versions, message 3966 is generated and the transaction is rolled back. If the shrinking process

Measurement	Description	Measurement Unit	Interpretation
			succeeds, space becomes available in **tempdb**. Otherwise tempdb runs out of space and the following occurs: • Write operations continue to execute but do not generate versions. An information message (3959) appears in the error log, but the transaction that writes data is not affected. • Transactions that attempt to access row versions that were not generated because of a tempdb full rollback terminate with an error 3958.
Version store size:	Indicates the amount of space in tempdb that is currently being used to store snapshot isolation level row versions.	KB	This information helps determine the amount of space needed in the **tempdb** database for the version store. Monitoring this measure over a period of time provides a useful estimate of additional space needed for **tempdb**.
Version generation rate:	Indicates the rate at which new row versions are added to the snapshot isolation version store in tempdb.	KB/Sec	The values of the *Version generation rate* and *Version cleanup rate* measures can be used to predict **tempdb** space requirements.
Version cleanup rate:	Indicates the rate at which row versions are removed from the snapshot isolation version store in tempdb.	KB/Sec	Once every minute, a background thread removes row versions that are no longer needed and frees up the version space in **tempdb** . A long- running transaction prevents space in the version store from being released if it meets any of the following conditions: • It uses row versioning- based isolation • It uses triggers, MARS, or online index build operations • It generates row versions

Measurement	Description	Measurement Unit	Interpretation
Version store units:	Indicates the number of allocation units currently active in the snapshot isolation version store in tempdb.	Number	
Version store units creation:	Indicates the number of new version store units created in the version store since the last measurement period.	Number	
Version store units deletion:	Indicates the number of version store units that were truncated since the last measurement period.	Number	A version store unit is truncated when SQL Server determines that none of the version rows stored in the version store unit are needed to run active transactions.
Update conflict ratio:	Indicates the percentage of transactions using the snapshot isolation level that have encountered update conflicts within the last second.	Percent	An update conflict occurs when a snapshot isolation level transaction attempts to modify a row that last was modified by another transaction that was not committed when the snapshot isolation level transaction started.
Longest transaction running time:	Indicates the length of time since the start of the transaction that has been active longer than any other current transaction.	Secs	Row versions are stored in **tempdb** for as long as an active transaction needs to access it. If the value of this measure is very high, then it indicates that a transaction has been running for an unreasonable period of time, and is thus preventing the database engine from freeing space from **tempdb**. If the *Free space in tempdb* measure touches alarmingly low levels, then you might have to identify the long running transaction and terminate it. Use *fn_transactions()* to identify the transaction.
Active transactions:	Indicates the number of	Number	This measure is a good indicator of the

Measurement	Description	Measurement Unit	Interpretation
	currently active transactions.		current workload on the database server.
Snapshot transactions:	Indicates the number of currently active transactions using the snapshot isolation level.	Number	The value of this measure changes when the first data access occurs, not when the *BEGIN TRANSACTION* statement is issued. Also, note that this measure does not include system transactions.
Update snapshot transactions:	Indicates the number of currently active transactions using the snapshot isolation level that perform update operations.	Number	The sum of *Update snapshot transactions* and *Non snapshot version transactions* represents the total number of transactions that participate in version generation. The difference of Snapshot transactions and Update snapshot transactions reports the number of read- only snapshot transactions.
Non snapshot version transactions:	Indicates the total number of currently active non-snapshot transactions that generate version records.	Number	
Temp tables creation rate:	Indicates the number of temporary tables/table variables created per second.	Tables/Sec	Temporary tables are created in **tempdb**. They are backed by physical disk and are even logged into the transaction log. They act like regular tables in that you can query their data via *SELECT* queries and modify their data via *UPDATE, INSERT* , and *DELETE* statements. If created inside a stored procedure they are destroyed upon completion of the stored procedure. Furthermore, the scope of any particular temporary table is the session in which it is created; meaning it is only visible to the current user. You can create indexes and statistics on temporary tables. You can also apply Data Definition Language (DDL) statements against temporary tables to add constraints, defaults, and referential

Measurement	Description	Measurement Unit	Interpretation
			integrity such as primary and foreign keys. You can also add and drop columns from temporary tables. The syntax for creating table variables is quite similar to creating either regular or temporary tables. The only differences involve a naming convention unique to variables in general, and the need to declare the table variable as you would any other local variable in Transact SQL. Unlike temporary or regular table objects, table variables have certain clear limitations. • Table variables can not have Non-Clustered Indexes • You can not create constraints in table variables • Statistics can not be created against table variables • You can not create default values on table variable columns Temporary tables are usually preferred over table variables for a few important reasons: they behave more like physical tables in respect to indexing and statistics creation and lifespan.

3.3.4 SQL User Processes Test

This test reports the number and state of sessions of each user who is currently connected to the Microsoft SQL server. Using the metrics reported by this test, administrators can promptly isolate idle sessions and suspended sessions, which are a drain on a server's resources.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each user currently connected to the Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC** roles in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDEUSER** - In the **EXCLUDEUSER** text box, specify a comma-separated list of user names that need to be excluded from monitoring. By default, *none* is displayed here indicating that this test monitors connections initiated by all current users to the Microsoft SQL server, by default.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
13. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the

detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

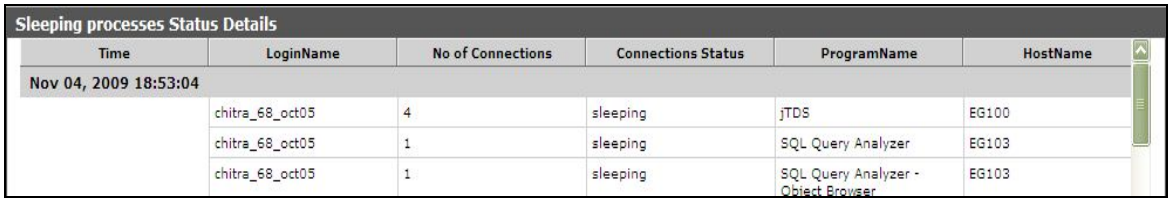
Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total processes:	Indicates the total number of sessions currently open on the server for this user.	Number	
Running processes:	Indicates the number of sessions of this user that are currently active.	Number	The detailed diagnosis of this measure, if enabled, will provide the complete details of the active sessions of a

Measurement	Description	Measurement Unit	Interpretation
			particular user. Using this information, you can understand how each of the connections were made - i.e., using which program - and from where - i.e., from which host.
Sleeping processes:	Indicates the number of sessions initiated by this user that are currently idle.	Number	Ideally, the value of this measure should be low. A high value is indicative of a large number of idle sessions, which in turn causes the unnecessary consumption of critical server resources. Idle sessions also unnecessarily lock connections from the connection pool, thereby denying other users access to the server for performing important tasks. The detailed diagnosis of this measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.
Suspended processes:	Indicates the number of sessions initiated by this user that are currently suspended.	Number	A session can switch to a suspended state if one/more processes triggered in that session could not continue executing; a possible reason for this could be that the processes are waiting for blocked rows or a blocked table to be released. The detailed diagnosis of this measure, if enabled, will provide the complete details of the suspended sessions of a particular user.
Background processes:	Indicates the number of background processes	Number	The detailed diagnosis of this measure, if enabled, provides the details

Measurement	Description	Measurement Unit	Interpretation
	currently running for this user.		pertaining to the background processes currently executing.

The detailed diagnosis of the *Sleeping processes* measure, if enabled, will provide the complete details of the idle sessions of a particular user. Using this information, you can understand how each of the idle connections were made - i.e., using which program - and from where - i.e., from which host.



Sleeping processes Status Details

Time	LoginName	No of Connections	Connections Status	ProgramName	HostName
Nov 04, 2009 18:53:04					
	chitra_68_oct05	4	sleeping	jTDS	EG100
	chitra_68_oct05	1	sleeping	SQL Query Analyzer	EG103
	chitra_68_oct05	1	sleeping	SQL Query Analyzer - Object Browser	EG103

Figure 3.21: The detailed diagnosis of the Sleeping processes measure

3.4 The MS SQL Databases Layer

The test associated with this layer (see Figure 3.22) monitors space usage on an MS SQL server database.

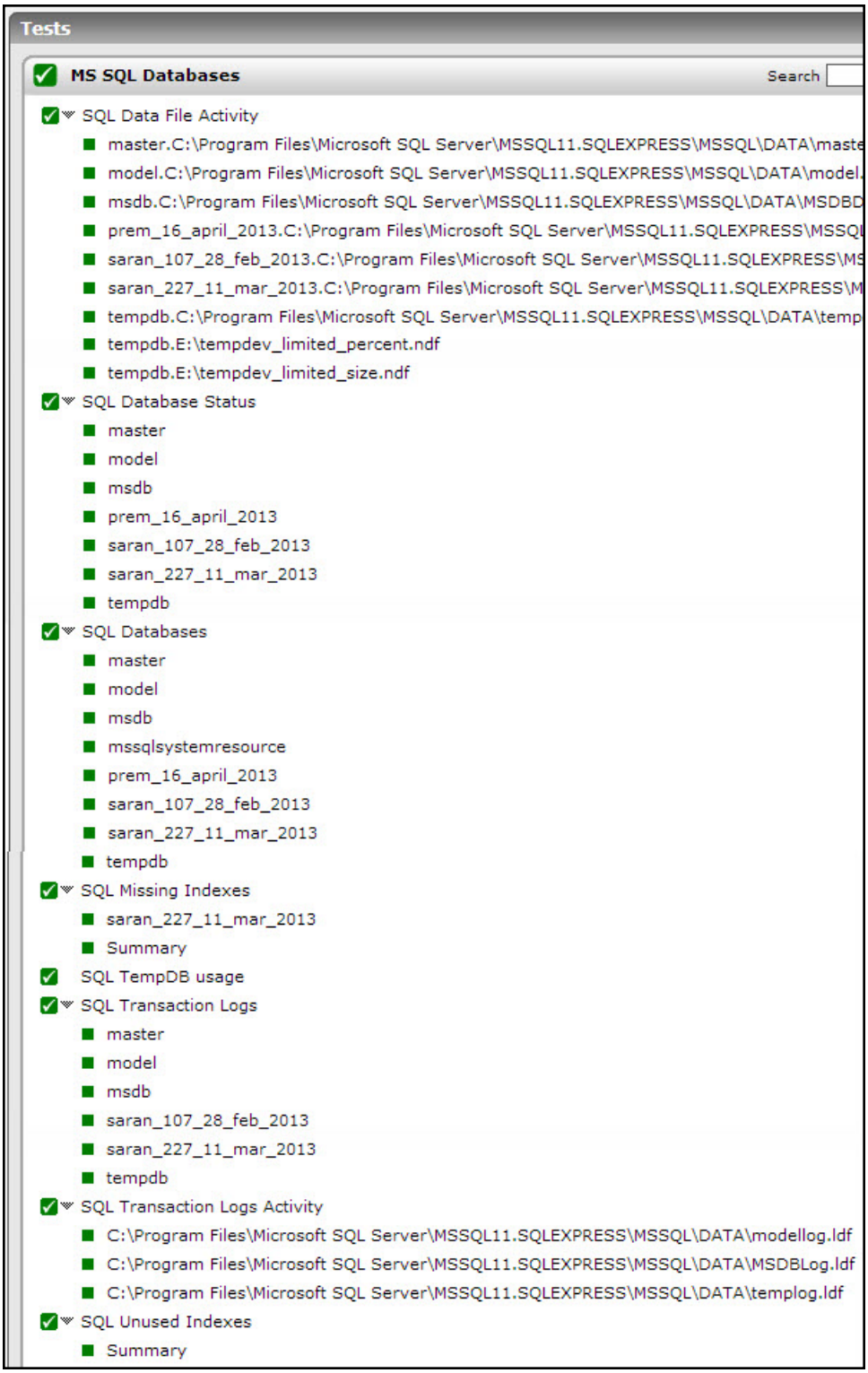


Figure 3.22: The tests associated with the MS SQL Databases Layer

3.4.1 SQL TempDB Usage Test

The *tempdb* system database is a global resource that is available to all users connected to the instance of SQL Server and is used to hold the following:

- Temporary user objects that are explicitly created, such as: global or local temporary tables, temporary stored procedures, table variables, or cursors.
- Internal objects that are created by the SQL Server Database Engine, for example, work tables to store intermediate results for spools or sorting.
- Row versions that are generated by data modification transactions in a database that uses read-committed using row versioning isolation or snapshot isolation transactions.
- Row versions that are generated by data modification transactions for features, such as: online index operations, Multiple Active Result Sets (MARS), and AFTER triggers.

Since *tempdb* is shared by multiple users/applications, if *tempdb* runs out of disk space, it can cause significant disruptions in the SQL Server production environment and can suspend operations of all applications that are using it. To prevent such an eventuality, you can use the **SQL TempDB Usage** test to continuously track tempDB usage, capture potential space contentions, and initiate measures to avert the contention.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **CONFIRM PASSWORD** - Confirm the password by retyping it.

8. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
9. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
10. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total size:	Indicates the total size of tempdb.	MB	
User objects:	Indicates the amount of space allocated from uniform extents to user objects in the tempdb database.	MB	User objects include both user-defined tables and indexes, and system catalog tables and indexes. User-defined tables include the global temporary tables such as ##t, and local temporary tables such as #t. Both of these objects are session scoped but a global temporary table lives until all sessions that are using it expire or terminate. Local temporary tables, on the other hand, are destroyed when the scope (for example, stored procedure or session) they were created in expires or terminates. Local temporary tables also include table variables such as @t, the return value in table valued functions, and the mapping index for online clustered index build with the *SORT_IN_TEMPDB* option. A high value of this measure indicates that a lot of space has been used by user objects.

Measurement	Description	Measurement Unit	Interpretation
			Compare the value of this measure with that of the Internal objects, Version store, and Mixed extent measures to know where the maximum tempdb space has been spent.
Internal objects:	Indicates the amount of space allocated from uniform extents to internal objects in the **tempdb** database.	MB	Internal objects are created internally by SQL Server. These objects are used: a. To store intermediate runs for sort. b. To store intermediate results for hash joins and hash aggregates. c. To store XML variables or other large object (LOB) data type variables. The LOB data type includes all of the large object types: text, image, ntext,varchar (max), varbinary(max), and all others. d. By queries that need a spool to store intermediate results. e. By keyset cursors to store the keys. f. By static cursors to store a query result. g. By Service Broker to store messages in transit. h. By INSTEAD OF triggers to store data for internal processing. Internal objects are also used by any feature that uses these operations. For example, DBCC CHECK internally uses a query that may need to spool intermediate results. Query notification and event notification use Service Broker, so they need space in **tempdb** as well. Compare the value of this measure with that of the *User objects, Version store*, and *Mixed extent* measures to know where the maximum tempdb space has been spent. If this measure reports the highest value it implies that query plans are making heavy

Measurement	Description	Measurement Unit	Interpretation
			use of the **tempdb**. This is not necessarily a problem, but you may want to look at the query plans to see if alternate query plans can be generated by creating indexes or by re-formulating the queries so as to minimize **tempdb** space usage.
Version store:	Indicates the amount of space allocated from uniform extents for the version store.	MB	Version stores are used to store row versions generated by transactions for features such as snapshot isolation, triggers, MARS (multiple active result sets), and online index build. There are two version stores in **tempdb** for the whole instance of SQL Server. The online index build version store is for row versions from tables that have online index build operations on them. The common version store is for row versions from all other tables in all databases. Compare the value of this measure with that of the User objects, Internal objects, and Mixed extent measures to know where the maximum tempdb space has been spent. If this comparative analysis reveals that this measure reports the highest value, it implies that version store cleanup cannot keep pace with version generation. See if a long-running transaction is preventing version store cleanup. Or, a high transaction throughput might be generating a large number of versions per minute. The background task cleans up versions every minute.
Mixed extent:	Indicates the amount of space that is used by objects of multiple types (user objects, internal	MB	Extents are the basic unit in which space is allocated to tables and indexes. To make its space allocation efficient, SQL Server does not allocate entire extents to

Measurement	Description	Measurement Unit	Interpretation
	objects, version store, Index Allocation Map (IAM) pages, etc.)		tables with small amounts of data. SQL Server has two types of extents - Uniform extents are owned by a single object; all eight pages in the extent can only be used by the owner object. A mixed extent is a single extent that contains multiple tables; it can be shared by up to eight objects. A high value indicates that a large amount of tempdb space is occupied by mixed extents. Compare the value of this measure with that of the User objects, Internal objects, and Version store measures to know where the maximum tempdb space has been spent.
Free space:	Indicates the amount of unused space in the tempdb database.	MB	A high value is desired for this measure.
Usage of allocated space:	Indicates the percentage of allocated tempdb space that is currently in use.	Percent	A consistent rise in the value of this measure could indicate a gradual, but steady erosion of space in the tempdb database. A value close to 100% signals a potential space crunch in the **tempdb** , which can affect the performance of all the applications using the **tempdb**. Under such circumstances, it would be good practice to compare the values of the *User objects, Internal objects, Version store*, and *Mixed extents* measures to know which object type is consuming the most space in the **tempdb**. You can then use the DMVs in the SQL server to analyze which Transact-SQL statements are the top consumers of tempdb space. You can kill such tasks, where appropriate, to free space.
Is auto- growth enabled?	Indicates whether/not		Auto-growth is the process by which the

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td>auto-growth is enabled for th tempdb database.</td><td></td><td>SQL Server engine expands the size of a database file when it runs out of space. The amount by which a database file grows is based on the settings that you have for the file growth options for your database.

If this setting is enabled for one/more database files in the tempdb database, this measure will report the value Yes. If this setting is not enabled for any of the database files in the tempdb database, then this measure will report the value No.

The numeric values that correspond to the above-mentioned measure values are listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>100</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:

Typically, this measure reports the Measure Values listed in the table above as the status of the Autogrowth setting. In the graph of this measure however, the status will be represented using the numeric values – i.e., 100 and 0.</td></tr>
<tr><td>Max file size:</td><td>Indicates the maximum size upto which the database files (for which the Auto-growth setting has been enabled) in the tempdb database can grow. This measure will be reported only if the ‘Is auto- growth enabled?’ measure returns the value ‘Yes’.</td><td>MB</td><td>There are three different settings you can use to identify how your database files will grow. They can grow by a specific size, a percentage of the current size, or not grow at all. Additionally you can set your files to unrestricted growth, which means they will keep growing as they need more space or you run out of disk space. Or you can restrict the growth of a database file to grow no larger than a specified size. Each one of these different auto-grow settings have defaults, or you can set them for each</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			database file. The default auto- growth settings for a database are rarely the ideal settings for how your database should grow. If you have an idea of the growth profile of your database when you first build it then you should set your auto- growth properties based on those growth projections. If you don't have any idea of how fast your database will grow then you should be monitoring for auto-growth events. Knowing how often your database grows will give you some ideas of the growth rate of your database.
Free disk space:	Indicates the amount of disk space that is currently available for use for tempdb database.	MB	A high value implies that the tempdb has adequate disk space for growth. If the value of this measure is low, then you may have to fine- tune your auto- growth settings accordingly.
Usage as % of max size:	Indicates the percentage of Max file size that is currently available for use for tempdb database. This measure will be reported only if the '*Is auto- growth enabled?*' measure returns the value 'Yes'.	Percent	This measure reports the *Free disk space* value as a percentage of Max file size. In other words: *Free disk space/Max file size * 100* If your database is set to auto-grow till disk capacity is reached, then, a high value of this measure indicates that there is enough space for the **tempdb** database to grow. A low value indicates that there is very little disk space for the use of **tempdb**. You may then have to free up some disk space or fine-tune your auto-growth settings.

3.4.2 SQL Databases Test

This test monitors the transactions that occur on every database of an Microsoft SQL server.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "**INSTANCE**" parameter.
5. **SSL** - By default, the **SSL** flag is set to **No**, indicating that the target Microsoft SQL server is not SSL-enabled by default. To enable the test to connect to an SSL-enabled Microsoft SQL server, set the **SSL** flag to **Yes**.
6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
7. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Active transactions:	This value indicates the current number of active transactions yet to be committed to the database.	Number	A large number for this value may indicate a large number of active transactions. Alternately this may also indicate that due to some reasons the users are not able to complete the transactions.
Transaction rate:	This measure indicates the number of transactions that are started for the database per second.	Trans/Sec	A high value of this metric indicates a lot of transactional activity happening to the database

Measurement	Description	Measurement Unit	Interpretation
Replication transaction rate:	This value indicates the number of transactions per second read out of the transaction log of the publication database and delivered to the distribution database.	Trans/Sec	A high value indicates that there is more replicated transactions happening from the publication database and is being sent to the distribution database.
Pending replication transactions:	Indicates the number of pending replication transactions in the database.	Number	This is the number of transactions in the transaction log of the publication database marked for replication, but not yet delivered to the distribution database.
Data file size:	This metric is the cumulative size of all the data files in the database server.	MB	The value of this metric provides an idea of the growth of the databases hosted by the database server.
Log flush waits:	This value indicates the number of transaction commits that are waiting in the log flush ready to be flushed.	Waits/Sec	A high value here may indicate non-optimal allocation of the log buffer related parameters.
Write transaction rate:	Indicates the number of transactions that wrote to the database and committed, in the last second.	Trans/Sec	

3.4.3 SQL Database Space Test

Typically, SQL Server databases have three types of files, as shown in the following table.

File	Description
Primary	The primary data file contains the startup information for the database and points to the other files in the database. User data and objects can be stored

File	Description
	in this file or in secondary data files. Every database has one primary data file. The recommended file name extension for primary data files is .mdf.
Secondary	Secondary data files are optional, are user-defined, and store user data. Secondary files can be used to spread data across multiple disks by putting each file on a different disk drive. Additionally, if a database exceeds the maximum size for a single Windows file, you can use secondary data files so the database can continue to grow. The recommended file name extension for secondary data files is .ndf.
Transaction Log	The transaction log files hold the log information that is used to recover the database. There must be at least one log file for each database. The recommended file name extension for transaction logs is .ldf.

All data files are stored in the filegroups listed in the following table.

Filegroup	Description
Primary	The filegroup that contains the primary file. All system tables are allocated to the primary filegroup.
User-defined	Any filegroup that is specifically created by the user when the user first creates or later modifies the database.

If even a single file group in a database runs out of free space, serious performance degradations will be noticed in applications that depend on that file group for their data needs. To avoid this, administrators must track space usage both at the database-level and at the individual file group-level and proactively identify those file groups that are over-utilized. The **SQL Database Space** test provides administrators with both these usage insights. By monitoring the space usage in each SQL database, the test points administrators to those databases that are consuming too much space and reveals the type of data that is hogging space – data in tables? Or indexes? Alongside, the test also reports usage metrics for each file group in every SQL database, and accurately pinpoints those file groups that may soon fill up the disk! This way, the test turns the spotlight on databases and file groups that may have to be resized to ensure peak application performance.

Note:

This test will report metrics for file groups only if Microsoft SQL Server 2008 R2 (and above) is monitored.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server database and every file group in each database

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.

4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – Provide the name of a SQL user with the **Sysadmin** role. However, if you do not want to expose the credentials of a **Sysadmin**, then create a special user for this purpose on each of the databases to be monitored, and make sure that you assign any of the following privileges to that user:
   - Assign the **db_datareader** privilege to that user in each of the databases to be monitored; (OR)
   - Assign the **PUBLIC** role to that user, and grant **execute** permission to that user for the **sp_spaceused** procedure in every database to be monitored

   Note that the name of the special user should be the same in all the databases.
7. **PASSWORD** – Provide the password of the specified **USER**.
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDEDB** - Specify a comma-separated . list of databases for which the space computation need not be done (e.g., *temp*). The default value is '*none*'.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
13. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total size:	The space allocated to a database.	MB	The *Total size* does not include the size of the database log files. **This measure is reported only for a database**.
Reserved space percent:	The percentage of allocated space reserved for tables and indexes of a database	Percent	If the value of this measure reaches 100%, it indicates that the total space in the database has been completely allocated. New tables/indexes can be added to the database, only if its total size is increased. **This measure is reported only for a database**.
Reserved space:	The amount of allocated space reserved for the tables and indexes created on a database	MB	If the value of this measure becomes equal to that of the *Total size* measure, new tables/indexes can no longer be created on the database. To create new tables, you must increase the database size. **This measure is reported only for a database.**
Data space:	The amount of allocated space used by data in tables.	MB	The total allocated space that is in use in a database is the sum of the value of the *Data space* and *Index space* measures. **This measure is reported only for a database**.
Index space:	The amount of allocated space used by indexes.	MB	
Unused space:	The amount of allocated space that is available for use in the database	MB	This is the difference between the *Total size* and *Reserved space.* **This measure is reported only for a database.**
Is auto- growth	Indicates whether/not auto-		**This measure is reported only for file**

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>enabled?</td><td>growth is enabled for this file group.</td><td></td><td>groups.

Auto-growth is the process by which the SQL Server engine expands the size of a database file when it runs out of space. The amount by which a database file grows is based on the settings that you have for the file growth options for the data file.

If this setting is enabled for even one file in a file group, this measure will report the value Yes. If this setting is not enabled for any of the database files in a file group, then this measure will report the value No.

The numeric values that correspond to the above-mentioned measure values are listed in the table below:
<table><tr><th>Measure Value</th><th>Numeric Value</th></tr><tr><td>Yes</td><td>100</td></tr><tr><td>No</td><td>0</td></tr></table>
Note:

Typically, this measure reports the Measure Values listed in the table above as the status of the Autogrowth setting. In the graph of this measure however, the status will be represented using the numeric values – i.e., 100 and 0.</td></tr>
<tr><td>Max file size:</td><td>Indicates the maximum size upto which the database files in this file group can grow.</td><td>MB</td><td>This measure is reported only for file groups.

Each database file that is associated with your database has an auto-growth setting. There are three different settings you can use to identify how your</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			database files will grow. They can grow by a specific size, a percentage of the current size, or not grow at all. Additionally you can set your files to unrestricted growth, which means they will keep growing as they need more space or you run out of disk space. Or you can restrict the growth of a database file to grow no larger than a specified size. Each one of these different auto-grow settings have defaults, or you can set them for each database file. If the auto-growth setting is not enabled at all for a file in a file group, then the amount of space that was originally allocated to that file will be regarded as the *Max file size* of that file. On the other hand, if the Auto-growth setting is enabled for a file in the file group, then the *Max file size* of that file will be one of the following: • If a specific size limit is explicitly set for the file, then this will be considered as the *Max file size* of that file. • If no size limit is set for the file, then the total capacity of the disk drive in which that file resides will be considered as the *Max file size* of that file. So, if a file group consists of a few data files for which auto-growth is enabled and a few others for which it is disabled, then the *Max file size* of that file group will be a sum total of the following: • The sum of the space allocated to each of the files for which auto-growth is not enabled;

Measurement	Description	Measurement Unit	Interpretation
			• The sum of the maximum size limits, if defined, for each file for which auto-growth is disabled; • The sum of the total capacity of the disks containing the auto- growth-enabled files for which no size limit is defined.
Free disk space:	Indicates the amount of disk space that is currently available for use for this file group.	MB	**This measure is reported only for file groups**. A high value implies that the database files in the file group have adequate disk space for growth. If the value of this measure is low, then you may have to fine- tune your auto- growth settings accordingly.
Free disk space percent:	Indicates the percentage of Max file size that is currently available for use for this file group.	Percent	**This measure is reported only for file groups**. This measure reports the *Free disk space* value as a percentage of Max file size. In other words: *Free disk space/Max file size * 100* If many files in a file group are set to auto-grow till disk capacity is reached, then, a high value of this measure indicates that there is enough space for the files to grow. A low value indicates that there is very little room for file growth.

3.4.4 SQL Data File Activity Test

By periodically monitoring the I/O activity on each datafile on the Microsoft SQL server and observing the growth in size of the datafile, this test sheds light on the following:

- Datafiles that are experiencing I/O bottlenecks;
- Datafiles that are consuming too much disk space

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABAS**e, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the databases on the Microsoft SQL server by default. To exclude specific databases from the monitoring scope of this test, provide a comma-separated list of databases in the **EXCLUDE INFO** text box.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

13. **SHOW DATAFILE PATH** - This test reports a set of results for each datafile on the target Oracle database server. This means that every datafile is a descriptor of this test. By default, while displaying the descriptors of this test, the eG monitoring console does not prefix the datafile names with the full path to the datafiles. This is why, the **SHOW DATAFILE PATH** flag is set to **No** by default. If you want the data file names to be prefixed by the full path to the data files, then, set the **SHOW DATAFILE PATH** flag to **Yes**.
14. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.
15. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Write rate:	Indicates the rate at which writes occurred on this datafile.	Writes/Sec	
Data write rate:	Indicates the rate at which data was written to this datafile.	KB/Sec	
I/O stall writes:	Indicates the total time taken to write to this datafile.	Millisecs	A high value for this measure could indicate a bottleneck while writing to the datafile. By comparing the value of this measure across datafiles, you can identify the data file to which write operations are taking too long to complete.

Measurement	Description	Measurement Unit	Interpretation
Read rate:	Indicates the rate of reads from this datafile.	Reads/Sec	
Data read rate:	Indicates the rate at which data was read from this datafile.	KB/Sec	
I/O stall reads:	Indicates the total time taken to read from this datafile.	Millisecs	A high value for this measure could indicate a bottleneck while reading from the datafile. By comparing the value of this measure across datafiles, you can identify the datafile to which read operations are taking too long to complete.
I/O stall:	Indicates the total time taken for I/O to complete on this datafile.	Millisecs	A high value for this measure could indicate an I/O bottleneck on this datafile.
Size on disk:	Indicates the total size on disk of each datafile.	Bytes	This measure is used to determine the growth of the datafile. A low value is desired for this measure. A very high value, or a consistent increase in this value may adversely impact I/O operations. You may want to consider maintaining multiple datafiles of smaller sizes to improve I/O efficiency, and to speed up backup/restore operations.

3.4.5 SQL Database Status Test

If a user complaints of problems while accessing a database, the knowledge of the current state of that database will enable administrators to promptly diagnose the reason for such an occurrence. This test auto-discovers all the databases on a Microsoft SQL server and reports the current state of each database, thereby enabling administrators to easily troubleshoot issues related to database access.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server database monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the INSTANCE parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the DOMAIN text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the DOMAIN can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the DOMAIN text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the USER name and PASSWORD that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

- The eG manager license should allow the detailed diagnosis capability
- Both the normal and abnormalfrequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td>Database status:</td><td>Indicates the current state of this database.</td><td></td><td>The states that can be reported by this measure, the numeric value that corresponds to each state, and a brief description of the state are provided below:
<table>
<tr><th>Value</th><th>State</th><th>Description</th></tr>
<tr><td>0</td><td>ONLINE</td><td>Database is available for access. The primary filegroup is online, although the undo phase of recovery may not have been completed.</td></tr>
<tr><td>1</td><td>RESTORING</td><td>One or more files of the primary filegroup are being restored, or one or more secondary files are being restored offline. The database is unavailable in this case.</td></tr>
<tr><td>2</td><td>RECOVERING</td><td>Database is being recovered. The</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>State</th><th>Description</th></tr>
<tr><td></td><td></td><td>recovering process is a transient state; the database will automatically become online if the recovery succeeds. If the recovery fails, the database will become suspect. The database is unavailable in this case.</td></tr>
<tr><td>3</td><td>RECOVERY_ PENDING</td><td>SQL Server has encountered a resource-related error during recovery. The database is not damaged, but files may be missing or system resource limitations may be preventing it from starting. The database is unavailable. Additional action by the user is required to resolve the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>State</th><th>Description</th></tr>
<tr><td></td><td></td><td>error and let the recovery process be completed.</td></tr>
<tr><td>4</td><td>SUSPECT</td><td>At least the primary filegroup is suspect and may be damaged. The database cannot be recovered during startup of SQL Server. The database is unavailable. Additional action by the user is required to resolve the problem.</td></tr>
<tr><td>5</td><td>EMERGENCY</td><td>User has changed the database and set the status to EMERGENCY. The database is in single- user mode and may be repaired or restored. The database is marked READ_ ONLY, logging</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>State</th><th>Description</th></tr>
<tr><td></td><td></td><td>is disabled, and access is limited to members of the sysadmin fixed server role. EMERGENCY is primarily used for troubleshooting purposes. For example, a database marked as suspect can be set to the EMERGENCY state. This could permit the system administrator read- only access to the database. Only members of the sysadmin fixed server role can set a database to the EMERGENCY state.</td></tr>
<tr><td>6</td><td>OFFLINE</td><td>Database is unavailable. A database becomes offline by explicit user action and</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Value</th><th>State</th><th>Description</th></tr>
<tr><td></td><td></td><td>remains offline until additional user action is taken. For example, the database may be taken offline in order to move a file to a new disk. The database is then brought back online after the move has been completed.</td></tr>
</table>
The detailed diagnosis of this measure reports the user access mode of the database, the database recovery model, and the log re-use wait state of the database.
</td></tr>
</table>

3.4.6 SQL Transaction Logs Test

Every SQL Server database has at least two files associated with it: one data file that houses the actual data and one transaction log file. The transaction log is a fundamental component of a database management system. All changes to application data in the database are recorded serially in the transaction log. The information recorded includes the following:

- the beginning time of each transaction
- the actual changes made to the data and enough information to undo the modifications made during each transaction (accomplished using before and after images of the data)
- the allocation and reallocation of database pages
- the actual commit or rollback of each transaction

Using this information, the DBMS can track which transaction made which changes to SQL Server data. However, it is only during transaction rollbacks, commits, and database recovery operations that the transaction log serves its true purpose. When a transaction is rolled back, the SQL Server copies before images to the database for every modification made since the *BEGIN TRANSACTION*. During a recovery

scenario you can use the transaction log to restore a database. This causes a roll forward of the transaction log. During a roll forward SQL Server will copy after images of each modification to the database. Using the logged data SQL Server ensures that each modification is applied in the same order that it originally occurred.

Lack of adequate space in the transaction log would hence have serious repercussions on the way the SQL server carries out database updations; sometimes, critical changes to the database could get lost due to a space crunch. The **SQL Transaction Logs** test enables administrators to constantly track the space consumption by the transaction log, so that administrators are instantly notified of inadequacies, and are prompted to act fast.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server database monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Log size:	Indicates the current size of the transaction log attached to this database.	MB	
Usage of allocated space:	Indicates the percentage of transaction log space that is currently in use.	Percent	Ideally, this value should be low. A high value or a value close to 100% requires immediate attention, as it indicates that the transaction log is suffering from severe space constraints. There is hence the danger of subsequent database modifications going unrecorded. Typically, excessive space usage can be attributed to too many changes been written to the log, but very little/none to the database. Further diagnosis can alone reveal the root-cause of this deviant behavior.
Is auto- growth enabled?	Indicates whether/not auto-growth is enabled for the transaction logs.		Turning on the **autogrowth** setting of the transaction logs enables the transaction log files to automatically grow by a configured percentage of the current log file size when more data space is required. If this setting is enabled for the transaction logs, this measure will report the value **Yes**. If this setting is not enabled, then this measure will report the value **No**. The numeric values that correspond to the above-mentioned measure values are listed in the table below:

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>Yes</td><td>100</td></tr>
<tr><td>No</td><td>0</td></tr>
</table>
Note:
Typically, this measure reports the Measure Values listed in the table above as the status of the Autogrowth setting. In the graph of this measure however, the status will be represented using the numeric values – i.e., 100 and 0.
This measure is available for Microsoft SQL Server 2008 R2 and above only.
</td></tr>
<tr><td>Auto-growth percent:</td><td>Indicates the percentage by which the transaction log files have been configured to automatically grow when more data space is required.</td><td>Percent</td><td>The growth increment of your transaction log must be large enough to stay ahead of the needs of your transaction units. Also, the growth increment must be large enough to avoid the following performance penalities:
a. If you run a transaction that requires more log space than is available, and you have turned on the autogrow option for the transaction log of that database, then the time it takes the transaction to complete will include the time it takes the transaction log to grow by the configured amount. If the growth increment is large or there is some other factor that causes it to take a long time, the query in which you open the transaction might fail because of a timeout error.
b. If you run a large transaction that requires the log to grow, other</td></tr>
</table>

Measurement	Description	Measurement Unit	Interpretation
			transactions that require a write to the transaction log will also have to wait until the grow operation completes. c. If you have many file growths in your log files, you may have an excessively large number of virtual log files (VLF). This can lead to performance problems with database startup/online operations, replication, mirroring, and change data capture (CDC). Additionally, this can sometimes cause performance problems with data modifications.
Auto growth size:	Indicates the size (in MB) by which the transaction log files have been configured to automatically grow when more data space is required.	MB	Besides considering the above factors when deciding on an **autogrowth** value to set, you also need to decide whether to set this value as a percentage or in MB. Typically, when you use percentage as an auto growth factor and the transaction log is smaller in size, you will probably encounter many repeated growth instances. At large file size or percentages you may encounter a timeout or long period of blocking while the file is grown. Typically, it is recommended to set the **AUTOGROW** value to an optimum value in MB instead of percentages. A general rule of thumb to you can use for testing is to set your **AUTOGROW** setting to about one-eight the size of the file. **These measures are available for Microsoft SQL Server 2008 R2 and above only**.
Max size of log:	Indicates the maximum size to which the	MB	You need to turn on the **MAXSIZE** setting for each file to prevent any one

Measurement	Description	Measurement Unit	Interpretation
	transaction log can grow.		file from growing to a point where it uses up all available disk space. **This measure is available for Microsoft SQL Server 2008 R2 and above only**.
Usage as % of max size:	Indicates the percentage of maximum log file size that is currently being used.	Percent	If the value of this measure keeps growing close to 100%, it indicates that the log is growing rapidly and may soon run out of space! In this case, do the following to make sure that the transaction log file does not become full: • In case of a database that has been configured with the Simple Recovery model, you need to ensure that the transaction log is automatically cleared on all checkpoints to ensure that there is adequate space to log subsequent transactions. You should also check for long- running transactions and search for possible flaws in the applications using the database. • In case of a database that has been configured with the Full Recovery model, check whether the transaction log backups have been defined and have succeeded. Also, verify whether the log backups occur often enough. Here again, keep your eyes open for long- running transactions and the reasons for their longevity. On the other hand, if the value of this measure touches 100%, it implies that the log file cannot grow more. An error 9002 is then raised. This means, that operations that change the state of the database cannot be successfully

Measurement	Description	Measurement Unit	Interpretation
			executed until sufficient free space is available in the transaction log. But, this also means that there's no guarantee that all the necessary information was recorded into the transaction log. The previous check list applies also this time but since the log is really full, you should also check the auto growth setting of the log file(s). Is it properly defined or should the log be let to grow more? Regardless of the decision to let the log grow more or not, the log can be truncated in a situation like this. **This measure is available for Microsoft SQL Server 2008 R2 and above only**.

3.4.7 SQL Missing Indexes Test

SQL Server allows you to put *indexes* on table columns, to speed up **WHERE** and **JOIN** statements on those columns. If a SQL query takes longer (much longer) to complete, it could be because one/more of these indexes are 'missing'. When the query optimizer optimizes a query, it identifies those *indexes* it would have liked to have used but were not available - these are called 'missing indexes'.

With the help of the **SQL Missing Indexes** test, you can be promptly alerted when the query optimizer finds one/more 'missing indexes'. Besides reporting the count of the missing indexes, the test also reveals which queries require these indexes, thus enabling you to quickly initiate index creation and query optimization.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the monitored Microsoft SQL server

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE,** and **PUBLIC ROLES** in this text box.
6. **PASSWORD** - The password of the specified **USER**
7. **CONFIRM PASSWORD** - Confirm the password by retyping it.
8. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "**INSTANCE"** parameter.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISPASSIVE –** If the value chosen is **Yes**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of Missing Indexes:	Indicates the total number of missing indexes found in the queries that are currently executing on this database.	Number	Use the detailed diagnosis of this measure to know which queries are missing one/more indexes. Knowledge of these queries can greatly aid database administrators in optimizing them, which in turn will increase the speed of database accesses and consequently, improve overall database performance.

3.4.8 SQL Unused Indexes Test

One of the balancing acts of SQL Server is the use of indexes. Too few indexes can cause scans to occur which hurts performance and too many indexes causes overhead for index maintenance during data updates and also a bloated database. Administrators hence need to continuously track which indexes are used and how they are used. *Unused indexes* in particular can have a negative impact on performance. This is because when the underlying table data is modified, the index may need to be updated also. This takes additional time and can even increase the probability of blocking. To avoid this, administrators need to rapidly isolate the unused indexes and drop them. The **Unused Indexes** test facilitates this. Using this test, administrators can quickly determine the number of unused indexes in each database, and also understand how badly that index impacts database performance.

This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the monitored Microsoft SQL server

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The host for which the test is to be configured
3. **PORT** - The port on which the server is listening
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.

5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **public roles** in this text box.
6. **PASSWORD** - The password of the specified **USER**
7. **CONFIRM PASSWORD** - Confirm the password by retyping it.
8. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the "**INSTANCE**" parameter.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE –** If the value chosen is **YES**, then the Oracle server under consideration is a passive server in an Oracle cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of Unused Indexes:	Indicates the total number of unused indexes currently found in this database	Number	Use the detailed diagnosis of this measure to know which indexes in a database are unused. The detailed diagnosis also reveals how often each such index has been updated, so that you can assess the unnecessary overheads that the database may have incurred in the process.

3.4.9 SQL Transaction Logs Activity Test

Every database created on an Microsoft SQL server is associated with a database file (.mdf) and a log data file (.ldf). All transactions to the database are logged in the log data file. The server will use the logged transactional information to restore the database after a crash, or when a transaction runs into issues and may have to be rolled back.

This test periodically monitors the I/O activity on and the growth in the size of the log file associated with each database. Using the metrics reported by this test, you can proactively detect bottlenecks while reading from or writing to the log file and excessive disk space usage by the log file.

By default, this test is disabled. This test is disabled by default. To enable the test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the monitored Microsoft SQL server

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.

5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the databases on the Microsoft SQL server by default. To exclude specific databases from the monitoring scope of this test, provide a comma-separated list of databases in the **EXCLUDE INFO** text box.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
13. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.
14. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Write rate:	Indicates the rate at which writes occurred on this log file.	Writes/Sec	
Data write rate:	Indicates the rate at which data was written to this log file.	KB/Sec	Ideally, the value for this measure should be low. If the value for this measure is high, use the detailed diagnosis of the Num of waits measure to identify the queries that are causing the waits to remain for a long time. You may want to finetune the queries to reduce wait time.
I/O stall writes:	Indicates the total time taken to write to this log file.	Millisecs	A high value for this measure could indicate a bottleneck while writing to the log file.By comparing the value of this measure across log files, you can identify the log file to which write operations are taking too long to complete.
Read rate:	Indicates the rate of reads from this log file.	Reads/Sec	
Data read rate:	Indicates the rate at which data was read from this log file.	KB/Sec	
I/O stall reads:	Indicates the time taken to read from this log file.	Millisecs	A high value for this measure could indicate a bottleneck while reading from the log file. By comparing the value of this measure across log files, you can identify the log file to which read operations are taking too long to complete.
I/O stall:	Indicates the total time	Millisecs	A high value for this measure could

Measurement	Description	Measurement Unit	Interpretation
	taken for I/O to complete on this log file.		indicate an I/O bottleneck on this log file.
Size on disk:	Indicates the total size on disk of each logfile.	Bytes	This measure is used to determine the growth of the logfile.

3.5 SQL Mirroring Status Test

Database mirroring is a solution for increasing the availability of a SQL Server database. Mirroring is implemented on a per-database basis and works only with databases that use the full recovery model.

Database mirroring maintains two copies of a single database that must reside on different server instances of SQL Server Database Engine. Typically, these server instances reside on computers in different locations. Starting database mirroring on a database, initiates a relationship, known as a *database mirroring session*, between these server instances.

One server instance serves the database to clients (the *principal server*). The other instance acts as a hot or warm standby server (the *mirror server*), depending on the configuration and state of the mirroring session. When a database mirroring session is synchronized, database mirroring provides a hot standby server that supports rapid failover without a loss of data from committed transactions. A Witness is an optional instance of SQL Server that enables the mirror server to recognize when to initiate an automatic failover. Unlike the two failover partners, the witness does not serve the database. Supporting automatic failover is the only role of the witness.

When the session is not synchronized, the mirror server is typically available as a warm standby server (with possible data loss).

It is hence evident that to prevent any data loss during failover, a database mirroring session should be in the *synchronized state* and a Witness should be up and running. If one or both the aforesaid conditions are not fulfilled, data loss is bound to occur. This is why, administrators should continuously track the state of every database mirroring session and witness on a SQL server instance. This is where the **SQL Mirroring Status** test helps.

For each database on a SQL server instance on which database mirroring is enabled, this test reports the current status of the mirroring session of that database, reveals what role that database plays in the mirroring session, and the current state of the witness. This way, administrators are promptly alerted when any mirroring session or witness switches to an abnormal state.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Mirroring state:	Indicates the mirroring state of this database mirroring session.		The values that this measure can report and their corresponding numeric values are as follows:

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>DISCONNECTED</td><td>1</td></tr>
<tr><td>SYNCHRONIZED</td><td>2</td></tr>
<tr><td>SYNCHRONIZING</td><td>3</td></tr>
<tr><td>PENDING_ FAILOVER</td><td>4</td></tr>
<tr><td>SUSPENDED</td><td>5</td></tr>
<tr><td>UNSYNCHRONIZED</td><td>6</td></tr>
<tr><td>SYNCHRONIZED</td><td>7</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the current status of a database mirroring session. In the graph of this measure however, the same is represented using the numeric equivalents only.
</td></tr>
<tr><td>Mirroring role state:</td><td>Indicates the role that is currently played by this database in the mirroring session.</td><td></td><td>The values that this measure can report and their corresponding numeric values are as follows:
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>PRINCIPAL</td><td>1</td></tr>
<tr><td>MIRROR</td><td>2</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the database role. In the graph of this measure however, the same is represented using the numeric equivalents only.
</td></tr>
<tr><td>Mirroring witness sate:</td><td>Indicates the current state of the witness in the database mirroring session.</td><td></td><td>The values that this measure can report and their corresponding numeric values are as follows:</td></tr>
</table>

<table>
<tr><th>Measurement</th><th>Description</th><th>Measurement Unit</th><th>Interpretation</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Measure Value</th><th>Numeric Value</th></tr>
<tr><td>UNKNOWN</td><td>1</td></tr>
<tr><td>CONNECTED</td><td>2</td></tr>
<tr><td>DISCONNECTED</td><td>3</td></tr>
</table>
Note:
By default, this measure reports the Measure Values listed in the table above to indicate the state of the witness. In the graph of this measure however, the same is represented using the numeric equivalents only.
</td></tr>
</table>

3.5.1 SQL Mirroring Transactions Test

The principal and mirror servers communicate and cooperate as *partners* in a *database mirroring session*. The two partners perform complementary roles in the session: the*principal role* and the *mirror role*. At any given time, one partner performs the principal role, and the other partner performs the mirror role. Each partner is described as *owning* its current role. The partner that owns the principal role is known as the *principal server*, and its copy of the database is the current principal database. The partner that owns the mirror role is known as the *mirror server*, and its copy of the database is the current mirror database. When database mirroring is deployed in a production environment, the principal database is the *production database*.

Database mirroring involves *redoing* every insert, update, and delete operation that occurs on the principal database onto the mirror database as quickly as possible. Redoing is accomplished by sending a stream of active transaction log records to the mirror server, which applies log records to the mirror database, in sequence, as quickly as possible. Unlike replication, which works at the logical level, database mirroring works at the level of the physical log record. Beginning in SQL Server 2008, the principal server compresses the stream of transaction log records before sending it to the mirror server. This log compression occurs in all mirroring sessions.

If transaction log records are not sent quickly by principal server or are not applied quickly by the mirror server, then the data in the principal and mirror databases will be out of sync; this will cause significant data loss during a failover. To avoid this, administrators must keep track of the log record traffic between the principal and mirror servers, proactively detect potential slowness in mirroring, figure out the probable source of the bottleneck, and clear it to ensure synchronization between the principal and mirror databases. This is where the **SQL Mirroring Transactions** test helps.

This test tracks the transactions started on a SQL server instance, measures the rate at which transaction log data is sent to the mirror server for synchronization, and the time taken by the mirror server to apply the data. In the process, the test pinpoints bottlenecks in database mirroring and where exactly the bottlenecks lie.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLE**s in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Desription	Measurement Unit	Interpretation
Queued log size:	Indicates the total number of kilobytes of log that have not yet been sent to the mirror server.	KB	A high value for this measure could indicate a bottleneck on the principal server or a network congestion obstructing data flow to the mirror server. Ideally, the value of this measure should be low.
Log bytes sent:	Indicates the rate at which log data was sent.	Bytes/sec	A high value is desired for this measure. A consistent drop in this value could indicate a processing bottleneck on the principal server.
Log compressed bytes sent:	Indicates the rate at which compressed bytes of log data was sent.	Bytes/Sec	
Log send flow control time:	Indicates the duration for which log stream messages waited for send flow control, in the last second.	Msecs	Sending log data and metadata to the mirroring partner is the most data-intensive operation in database mirroring and might monopolize the database mirroring and Service Broker send buffers. Use this counter to monitor the use of this buffer by the database mirroring session. A high value of this measure indicates that the queue in the actual layer sending the messages on the network is full. Hence this would indicate a network issue.
Transactions:	Indicates the rate at which transactions were started for the database.	Transactions/Sec	
Transaction delay:	Indicates delay in waiting for unterminated commit acknowledgment.	Msecs	High values in this counter can be a clear indicator of a bottleneck that is affecting performance and that end

Measurement	Desription	Measurement Unit	Interpretation
			users are seeing a delay in their transactions.
Avg transaction delay:	Indicates ratio between transaction delay and transaction per sec.	Msecs/transaction	
Log harden time:	Indicates the time for which log blocks waited to be hardened to disk, in the last second.	Msecs	If transactions are not hardened on the log drive on the mirror fast enough and you are using high safety, the principal might have to wait for the mirror to acknowledge hardening of log records before transactions can commit, resulting in degraded performance. A high value for this measure is therefore a cause for concern.
Log bytes received:	Indicates the rate at which log bytes were received.	Bytes/Sec	A high value is desired for this measure. A consistent drop in this value could indicate a processing bottleneck on the mirror server.
Log compressed bytes received:	Indicates the rate at which compressed log data was received.	Bytes/Sec	
Redo bytes:	Indicates the number of bytes of log rolled forward on te mirror database per second.	Bytes/Sec	
Redo queue:	Indicates the total number of kilobytes of hardened log that currently remain to be applied to the mirror database to roll it forward. This is sent to the Principal from the Mirror.	KB	A low value is ideal for this measure.

Measurement	Desription	Measurement Unit	Interpretation
Send/Receive ack time:	Indicates the time for which messages waited for acknowledgment from the partner, in the last second.	Msecs	This counter is helpful in troubleshooting a problem that might be caused by a network bottleneck, such as unexplained failovers, a large send queue, or high transaction latency. In such cases, you can analyze the value of this counter to determine whether the network is causing the problem.
Mirrored write transactions:	Indicates the number of transactions that wrote to the mirrored database and waited for the log to be sent to the mirror in order to commit, in the last second. This counter is incremented only when the principal server is actively sending log records to the mirror server.	Transactions/Sec	This counter is incremented only when the principal server is actively sending log records to the mirror server.

3.6 The Microsoft SQL Service Layer

The Microsoft SQL Service layer tracks the health of the services associated with an Microsoft SQL server (see Figure 3.23). The following sections provide more information on these tests and the measures reported by them.

Figure 3.23: Tests mapping to the Microsoft SQL Service layer

3.6.1 SQL Long Running Queries Test

This test reports the time taken by each database for executing queries, and also reveals which query is taking too long to execute. This way, resource-intensive queries to a database can be quickly isolated.

One set of results for each database on the Microsoft SQL server monitored

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each database on the Microsoft SQL server monitored

Note:

This test will execute only on Microsoft SQL Server 2005 (or above).

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the INSTANCE parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL**, **VIEW SERVER STATE**, **VIEW ANY DEFINITION**, **VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDEDB** - By default, the **EXCLUDEDB** text box will be set to a comma-separated list of databases that will be excluded from the monitoring scope of this test by default. You can append more databases to this list to exclude more databases, or can remove one/more databases from the list to include them in the monitoring purview.
11. **TOP N QUERY** - By default, the number ‘10’ is displayed in the **TOP N QUERY** text box. This implies that, by default, the detailed diagnosis of this test will display the top-10 queries to the database, in

terms of resource usage. If you want the detailed diagnosis to display more or less number of top queries, set the **TOP N QUERY** parameter to a number of your choice.

12. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
13. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
14. **AVG ELAPSED PER EXEC** - By default, to compute the *Avg elapsed time* and *Max elapsed time* of queries, this test considers the execution time of only those queries that have been running for over 10 seconds. Also, by default, the detailed diagnosis of this test will provide the details of only those queries that have been running for over 10 seconds. This is why, the **AVG ELAPSED PER EXEC** parameter is set to **10**, by default. You can change this default setting, if required.
15. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurement made by the test

Measurement	Description	Measurement Unit	Interpretation
Avg elapsed time:	Indicates the average time taken by queries to execute on this database.	Secs	If the value of this measure is very high, it could either indicate that the database is unable to process the queries quickly or that one/more queries to the database are taking too long to execute. Improper indexing and fragmented tables in the database are common causes for slowdowns at the database- level. Besides the above, queries that are

Measurement	Description	Measurement Unit	Interpretation
			improperly structured can also take time to execute. The longer a query executes on the database, higher would be the resource consumption of that query. It is therefore imperative that such resource-intensive queries are quickly isolated and fine-tuned, so as to prevent degradations in the performance of the database server. Using the detailed diagnosis of this measure, you can rapidly identify the resource-intensive queries to the database.
Max elapsed time:	Indicates the maximum time taken by the queries to this database.	Secs	

3.6.2 SQL Network Test

This test monitors the availability and response time from clients by an Microsoft SQL database server from an external perspective.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** – How often should the test be executed
2. **HOST**– The IP address of the Microsoft SQL server.
3. **PORT** – The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL user with the

Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.

7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **DATABASE** - The name of the database to connect to. The default is "master". To monitor multiple databases, ensure that the database names are provided as a colon-separated list. Alternatively, you can use the semi-colon as the separator for the database names.
11. **QUERY** – The select query to execute. The default is "select * from master.dbo.spt_monitor". If the target Microsoft SQL database server is installed as case sensitive, then the value of query parameter must be case sensitive. If multiple databases are specified in the **DATABASE** text box, then you will have to provide multiple queries here separated by a semi-colon (;) - for eg., *select * from master.dbo.spt_ monitor;select * from alarm*. Every **DATABASE** being monitored, should have a corresponding **QUERY** specification.
12. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
13. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
SQL server availability:	Indicates the availability of the server.	Percent	The availability is 100% when the server is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database server, or because the server has not been started. The availability is

Measurement	Description	Measurement Unit	Interpretation
			100% when the instance is responding to a request and 0% when it is not. Availability problems may be caused by a misconfiguration/malfunctioning of the database instance, or because the instance is using an invalid user account. Besides the above, this measure will report that the server is unavailable even if a connection to the database instance is unavailable, or if a query to the database fails. In this case, you can check the values of the DB connection availability and Query processor availability measures to know what is exactly causing the database instance to not respond to requests - is it owing to a connection unavailability? or is it due to a query failure?
SQL response time:	The time taken by the database to respond to a user query. This is the sum total of the connection time and query execution time.	Seconds	A sudden increase in response time is indicative of a bottleneck at the database server.
DB connection availability:	Indicates whether the database connection is available or not.	Percent	If this measure reports the value 100 , it indicates that the database connection is available. The value 0 on the other hand indicates that the database connection is unavailable. A connection to the database may be unavailable if the database is down or if the database is listening on a port other than the one configured for it in the eG manager or owing to a poor network link. If the SQL availability measure reports the value 0, then, you can check the value of this measure to determine whether/not it is due to the unavailability of

Measurement	Description	Measurement Unit	Interpretation
			a connection to the server.
Query processor availability:	Indicates whether the database query is executed successfully or not.	Percent	If this measure reports the value 100, it indicates that the query executed successfully. The value 0 on the other hand indicates that the query failed. In the event that the SQL availability measure reports the value 0, check the value of this measure to figure out whether the failed query is the reason why that measure reported a server unavailability.
Connection time to database server:	Indicates the time taken by the database connection.	Seconds	A high value could indicate a connection bottleneck. Whenever the SQL response time of the measure soars, you may want to check the value of this measure to determine whether a connection latency is causing the poor responsiveness of the server.
Query execution time:	Indicates the time taken for query execution.	Seconds	A high value could indicate that one/more queries to the database are taking too long to execute. Inefficient/badly designed queries to the database often run for long periods. If the value of this measure is higher than that of the Connection time measure, you can be rest assured that long running queries are the ones causing the responsiveness of the server to suffer.
Records fetched:	Indicates the number of records fetched from the database.	Number	The value 0 indicates that no records are fetched from the database

3.6.3 SQL Sessions Test

This test monitors the availability of the Microsoft SQL server from an internal perspective. This test returns measurements like Login/Sec, Logout/Sec, and number of user connections.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
6. **PASSWORD** - The password of the specified **USER**
7. **CONFIRM PASSWORD** - Confirm the password by retyping it.
8. **INSTANCE** – The name of a specific Microsoft SQL instance to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the “**INSTANCE**” parameter.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Logins:	This value indicates the total number of logins per second.	Logins/Second	A high value here indicates an increase in the rate of user logins into the SQL server. An unusual increase in the login rate may be an indicator of abnormal activity of database applications.
Logouts:	Indicates the total number of logouts per second.	Logouts/Second	A high value here indicates increase in rate of users logging out of SQL server. An unusually large number of logins and logouts can occur due to application retries being caused by errors during database access.
Current connections:	Indicates the number user connections to the server at an instant.	Number	As each user connection consumes some memory, a large number of user connections could affect throughput. By tracking the history of user connections, a database administrator can set the maximum expected number of concurrent users accordingly.
Logical connections:	Indicates the number of logical connections to the server.	Number	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

3.6.4 SQL Backup Details Test

To prevent the data loss that may occur due to the sudden failure of a SQL database, administrators are often advised to schedule the automatic backup of the databases on the Microsoft SQL server. It is recommended that these backup jobs are scheduled to occur frequently (say, once a day), so that the backup is always in sync with the data backed up. Failed or unusually fast backup jobs should also be detected quickly and marked for closer scrutiny, as this can cause data non-sync and increase the risk of data loss when disaster strikes. This is why, administrators will find the **SQL Backup Details** test very helpful! This test monitors the backup jobs configured for every SQL database. In the process, the test reports the type of backup that is configured, when the last backup job ran, and how long it took. This way, administrators can quickly identify databases that are not backed up as frequently as they would like them to be, rapidly detect backup jobs that may have failed to run as per schedule, and can pinpoint those databases where the last backup was suspiciously fast. With the help of these inferences, administrators can fine-tune backup schedules and can troubleshoot backup failures.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database on the MS SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the MS SQL server.
3. **PORT** - The port number through which the MS SQL server communicates. The default port is 1433.
4. **SSL** – If the MS SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
6. **PASSWORD** - The password of the specified **USER**
7. **CONFIRM PASSWORD** - Confirm the password by retyping it.
8. **INSTANCE** – The name of a specific MS SQL instance to be monitored. The default value of this parameter is “default”. To monitor an MS SQL instance named “CFS”, enter this as the value of the “instance” parameter.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed MS SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.

10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the MS SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
12. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Backup type:	Indicates the type of backup that is configured for this database.		The values that this measure can report and their corresponding numeric values are listed below: Measure Value / Numeric Value Full – 1 Diff-Database – 2 T-Log – 3 File-File Group – 4 Diff-File – 5 Partial – 6 Diff-Partial – 7 **Note:** By default, this measure will report one of the **Measure Value**s listed in the table

Measurement	Description	Measurement Unit	Interpretation
			above to indicate the backup type of a database. However, in the graph of this measure, the same is represented using the numeric equivalents only. You can use the detailed diagnosis of this measure to know the start time and end time of the backup operation, the time taken for the last backup operation, the user who has initiated the backup operation and name of the server in which the backup operation is performed.
Time taken for last backup operation:	Indicates the time it took for the last backup operation on this database to complete.	Minutes	If the value of this measure is very low for a database, it warrants a probe, as it could be owing to the failure of the backup job.
Interval since last backup:	Indicates the number of days that have elapsed since this database was last backed up.	Days	If the value of this measure is over 1 day, it could imply one of the following: • The database is not been backed up regularly, or; • The last backup job on the database failed Either way, further investigation may be required to determine the real reasons.

3.6.5 SQL Table Size Test

When faced with a disk space crunch on their critical SQL servers, administrators may want to know which databases are hogging the disk space, and which tables on each database have grown beyond permissible limits. The **SQL Table Size** test provides administrators with this information. The test auto-discovers the databases with tables that exceed a configured size limit, and reports the count of such 'large sized tables' for each database. You can use the detailed diagnosis of the test to know which tables in a database are of a large size.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**,

Performance as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Note:

This test is applicable only to Microsoft SQL Server 2005 (and above).

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database containing tables that are of size greater than the configured **TABLE SIZE GB**

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **TABLE SIZE GB** - The test will report the count of those tables that are of a size greater than the value (in GB) specified here.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
13. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation	
Table Count:	Indicates the number of tables in this database that are currently of a size greater than the value (in GB) configured against table size gb.	Number	Use the detailed diagnosis of this measure to know which tables in a database are consuming the maximum disk space.	

3.6.6 SQL Query Wait Activity Test

This test monitors the wait types on the Microsoft SQL server, and reports the number and duration of waits of each type.

This test has been disabled by default for this layer. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Note:

This test is applicable only to Microsoft SQL Server 2005 (and above).

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each wait type on the Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the wait types active on the Microsoft SQL server by default. To exclude specific wait types from the monitoring scope of this test, provide a comma-separated list of wait types in the **EXCLUDE INFO** text box.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
13. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.

14. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of waits:	Indicates the total number of waits for this wait type.	Number	The detailed diagnosis of this measure provides the query that caused the waits; you may want to fine-tune the queries to reduce the number of waits.
Avg wait time:	Indicates the average wait time for this wait type.	Seconds	Ideally, the value for this measure should be low. If the value for this measure is high, use the detailed diagnosis of the *Num of waits* measure to identify the queries that are causing the waits to remain for a long time. You may want to fine-tune the queries to reduce wait time.
Max wait time:	Indicates the maximum wait time for this wait type.	Seconds	Comparing the value of this measure across wait types will enable you to accurately isolate the wait type that is responsible for the longest waits.

3.6.7 SQL Session Activity Test

This test monitors the sessions initiated by each application on the Microsoft SQL server, and reports the level of activity each application has imposed on the server. This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this

test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for each application using the Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the Sysadmin role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDE INFO** - By default, this is set to none, indicating that the test will monitor all the applications that are using the Microsoft SQL server by default. To exclude specific applications from the monitoring scope of this test, provide a comma-separated list of applications in the EXCLUDE INFO text box.
11. **TOP-N** - By default, this is set to 10, indicating that the detailed diagnosis of this test will list only the top-10 sessions based on the elapsed time of the sessions. To view more or less number of top-n records in the detailed diagnosis, provide any other number of your choice in the **TOP-N** text box.
12. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the

ISNTLMV2 flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.

13. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
14. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.
15. **DETAILED DIAGNOSIS** - To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enabled/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of processes:	Indicates the total number of processes executed currently by this application on the server.	Number	The detailed diagnosis of this measure is used to view the complete details of the session. This includes the session ID, the user who has initiated the session, the current status of the session, the elapsed time, CPU time, open cursor count,active request count,blocking/blocked request count, the count of disk reads and writes, and the count of logical reads performed by the session. These information enables you to know how resource intensive the session is and how much I/O load has been generated by the session.
Open transaction count:	Indicates the total number of transaction executed	Number	In terms of a server overload, you can compare the value of this measure

Measurement	Description	Measurement Unit	Interpretation
	currently by the application on the server.		across the applications to identify the sessions that are responsible for increasing the transaction load on the server.
Open cursor count:	Indicates the total number of cursors executed currently by the application on the server.	Number	In terms of a server overload, you can compare the value of this measure across the applications to identify the sessions that are responsible for increasing the transaction load on the server.
Active request count:	Indicates the rate at which the application is sending requests to the server.	Reqs/Sec	
Blocking request count:	Indicates the number of requests of this application that are currently being blocked.	Number	Ideally, the value of this measure should be 0. If the measure reports a non-zero value, then you can use the detailed diagnosis of the Blocked processes measure of the *SQL System Processes* test to identify the exact query that is responsible for the blocking.
Open resultset count:	Indicates the number of resultsets that are currently open on the server for this application.	Number	
Blocked request count:	Indicates the number of requests of the application that are currently blocking.	Number	Ideally, the value of this measure should be 0. If the measure reports a non-zero value, then you can use the detailed diagnosis of the Blocked processes measure of the *SQL System Processes* test to identify the exact query that is responsible for the blocking.

3.6.8 SQL Error Log Test

This test reports the number and type of errors logged in the SQL server error logs. This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server 2005 (or above)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Microsoft SQL server being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **FILEPATH** - Enter the full path to the log file to be monitored.
5. **ISUTF16** - If the error log file to be monitored is encoded with UTF-16, then, set the **ISUTF16** flag to **Yes**. By default, this flag is set to **No**.
6. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
7. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. For instance, if you set to 1:1, it means that detailed measures will be generated every time this test runs, and also every time the test detects a problem.
8. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

   The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

   - The eG manager license should allow the detailed diagnosis capability
   - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
DeadLocks:	Indicates the number of deadlocks on the Microsoft SQL server since the last measurement period.	Number	The SQL error log will capture deadlock conditions only if trace is enabled on the SQL server. This means that if the value of this measure is 0, it could imply one of the following: • Tracing is not enabled, and therefore, the SQL error log is not able to capture any deadlock conditions; if you want, you can enable tracing by runing the following command from the SQL prompt: *DBCC TRACEON(1222,-1)* • Tracing is enabled, but no deadlock has occurred. This is a sign of good health. If this measure reports a non-zero value on the other hand, it is a cause for concern, as it indicates that one/more deadlocks have occurred. In this case, you can use the detailed diagnosis of this test to know more information about the deadlocks.
Informational messages:	Indicates the number of informational messages that were captured by the error log during the last measurement period.	Number	Messages with a severity level of 0 to 10 are informational messages and not actual errors.
User errors:	Indicates the number of user errors that were captured by the error log during the last measurement period.	Number	The value of this measure indicates the number of errors with a severity level between 11 and 16. Such errors are generated as a result of user problems and can be fixed by the user.
Software errors:	Indicates the number of	Number	All errors with severity levels 17 to 19

Measurement	Description	Measurement Unit	Interpretation
	software errors captured by the error log during the last measurement period.		will be counted as software errors. Severity level 17 indicates that SQL Server has run out of a configurable resource, such as locks. Severity error 17 can be corrected by the DBA, and in some cases, by the database owner. Severity level 18 messages indicate non- fatal internal software problems. Severity level 19 indicates that a nonconfigurable resource limit has been exceeded.
Fatal or system errors:	Indicates the number of fatal or system errors experienced by the target Microsoft SQL server during the last measurement period.	Number	Errors with severity levels 20 to 25 are typically categorized as fatal/system. Severity level 20 indicates a problem with a statement issued by the current process. Severity level 21 indicates that SQL Server has encountered a problem that affects all the processes in a database. Severity level 22 means a table or index has been damaged. To try to determine the extent of the problem, stop and restart SQL Server. If the problem is in the cache and not on the disk, the restart corrects the problem. Otherwise, use DBCC to determine the extent of the damage and the required action to take. Severity level 23 indicates a suspect database. To determine the extent of the damage and the proper action to take, use the DBCC commands. Severity level 24 indicates a hardware problem. Severity level 25 indicates some type of system error.
Warning messages	Indicates the number of warning messages found in the SQL error logs during the last measurement		

Measurement	Description	Measurement Unit	Interpretation
	period.		
Other errors:	Indicates the number of other errors captured in the Microsoft SQL server during the last measurement period.	Number	

3.6.9 SQL Job Details Test

This test measures the status of all the jobs executing on an Microsoft SQL server. This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for the Microsoft SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.

9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
11. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.
12. **DDFORSUCCESS** - By default, detailed diagnosis information will be available for the successful, failed and cancelled jobs reported by this test. Accordingly, the **DDFORSUCCESS** test flag is set to **Yes** by default. If you do not want detailed diagnosis for the successful jobs measures then set this flag to **No**.
13. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Total SQL jobs:	Indicates the total number of jobs that are currently running.	Number	
Successful SQL jobs:	Indicates the jobs that have been successfully executed.	Number	

Measurement	Description	Measurement Unit	Interpretation
Failed SQL jobs:	Indicates the jobs that have not been successfully executed.	Number	Ideally, the value for this measure should be zero. On other hand, if this measure reports a non-zero value, then you can use detailed diagnosis to identify the name of the job that has failed, the server name and the step at which the job failed.
Cancelled SQL jobs:	Indicates the jobs that were terminated by user interference.	Number	Ideally, the value for this measure should be zero. However, if this measure reports a non-zero value, then you can use detailed diagnosis to identify the name of the job that has been cancelled, server name and the step at which the job was cancelled.

3.6.10 SQL Job Status Test

This test reports the current status of configured SQL jobs. This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every job that has been configured for monitoring

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance

named “CFS”, enter this as the value of the **INSTANCE** parameter.

6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **JOBNAME** - Specify the job to be monitored in the **JOBNAME** text box. If multiple jobs are to be monitored, then provide a comma-seperated list of job names.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 (“NTLMv2”) was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as “Not applicable’ by the agent if the server is not up.
13. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of run times for the SQL job:	Indicates the number of times the job has run.	Number	
Avg duration of SQL job:	Indicates the average time taken by the job to execute.	Secs	By comparing the value of this metric across the current jobs, you can accurately identify the job that is taking too long to complete. Also, note that if the value of the *Number of run times for the SQL job* measure is equal to zero, then the *Avg duration of the SQL job* measure will not appear in the eG user interface.
Status of the SQL job:	Indicates the status of this job.	Number	If the value of this measure is 0, it means that the job has failed. In such a case, use the detailed diagnosis of the *Number of run times for the SQL job* measure to determine the root-cause of the failure. The value 3 for this measure indicates that the job was cancelled, and the value 1 denotes that the job executed successfully. However, note that if the value of the *Number of run times for the SQL* job measure is equal to zero, then the *Status of the SQL job* measure will not appear in the eG user interface.

3.6.11 SQL Applications Test

Sometimes the database performs poorly due, not to blocking, but to particularly heavy loads. Often the DBA will determine that the database simply cannot support the work that it is being asked to do and maintain adequate performance. This does not necessarily mean it is time to create more indexes or throw more hardware at the problem. One cannot always assume that periods of high utilization represent legitimate work. There could be problems in the applications that are running, or even problems caused by the user. Maybe the application has a data paging functionality, but the user has opted to receive the entire 100,000 row DataSet every time, even though she/he has applied a sort which gives her the one row she needs first with each

query. Regardless, it is important to identify performance issues and eliminate them. The **SQL Applications** test identifies which program has more connections open to (i.e., processes running in) the SQL database. Simply looking at the CPU cycles taken up by a process will not indicate which of these processes has been most active recently. For example, the SQL Server internal processes may have been running for days and will probably always show up as the processes that have taken the most CPU time since the database booted up. Hence, it is more helpful to find processes that have used lots of CPU for the majority of the time that they have been connected. This value represents how "expensive" a process is with respect to the SQL database server.

For each program that connects to the database server, the **SQL Applications** test reports the total CPU cycles for each second that the program is connected to the database. This value, represented by the *CPU cycles rate* measure, is an aggregate of all the CPU cycles consumed by every instance of the program while it is connected to the database server. The *Avg CPU cycles rate* measure represents the *CPU cycles rate* averaged across the number of processes in the database for the program under consideration. The *Avg CPU cycles rate* quantifies how bad a program is compared to the others, by dividing the *CPU cycles rate* by the number of connected instances. A high value for this value would indicate that every instance of the program was CPU-intensive. A lower value would indicate that the program may have some instances that cause performance problems, but also has instances that are mostly idle.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Note:

This test is applicable only to Microsoft SQL Server 2005 (and above).

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every program on the MS SQL server monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the

name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.

7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **EXCLUDEPATTERN** - Provide a comma-separated list of programs/processes on the SQL server that need to be excluded from monitoring. The default value is *none*, indicating that all processes are monitored by default. To make sure that the test ignores a few processes, specify the process names as a comma-separated list. For example: *SQL_Query_Analyzer,jTDS*. You can also use wild card patterns in your specification - for instance, *SQL*,*TDS,Microsoft**.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
13. **DD FREQUENCY** - Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *2:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against **DD FREQUENCY**.
14. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Number of processes:	Indicates the number of database processes associated with a specific program.	Number	A comparison of this value across programs will indicate which program is initiating most connections to the database. Comparison of this value over time can provide indications of potential changes in database activity characteristics of a program.
CPU cycles rate:	Indicates the number of CPU cycles consumed by all processes of a program, per minute of login.	Cycles/sec	The higher the value, the more CPU resources that the program is taking in the database.
Avg CPU cycles rate:	Indicates the number of CPU cycles consumed by a process of a program, per minute of login.	Cycles/Conn	This value is the ratio of the CPU cycles rate to the number of processes for a program.

The detailed diagnosis of the *CPU cycles rate* measure of this test provides details of the most expensive queries to the database - i.e., what host is a program running from, who is running it, and what application is running it, which database the program is accessing, etc (see Figure 3.24).

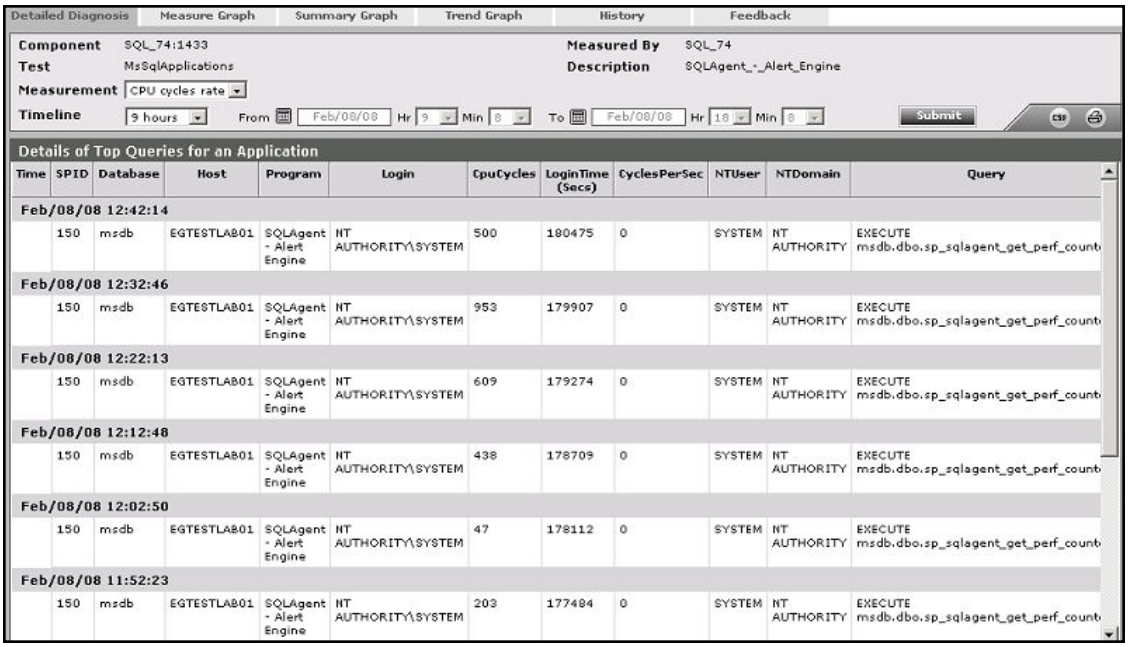


Figure 3.24: The detailed diagnosis of the CPU cycles rate measure

3.6.12 SQL Waits Test

This test reports key statistics pertaining to wait status. This test is specific to Microsoft SQL Server 2005 (or above), and will hence not report any measure for any of the other versions of the Microsoft SQL server.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server 2005 (or above)

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every type of wait on the Microsoft SQL Server 2005 (or above) being monitored.

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be

monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.

6. **USEPERFMON** – By default, this flag is set to **Yes**, indicating that this test uses the Windows Perfmon utility by default to pull out the metrics of interest. To instruct the test to use queries for metrics collection and not Perfmon, set this flag to **No**. Typically, when monitoring a Microsoft SQL server in an agent-based manner, its best to go with the default setting – i.e., use Perfmon for metrics collection. However, when monitoring the Microsoft SQL server in an agentless manner, its ideal to use queries instead of Perfmon to collect the required metrics. In such cases, set this flag to **No**.
7. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable" by the agent if the server is not up.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Avg wait time:	Indicates the average duration of this wait type.	Seconds	If a particular wait type is found to have persisted for a long time, it could indicate a processing overhead.
Waits in progress:	Indicates the number of processes currently waiting on this wait type.	Number	Closely monitoring *Waits in progress* along with *Avg wait time* over a period of time will reveal wait types that are locking critical system resources.
Waits started:	Indicates the number of waits started per second of this wait type.	Waits/sec	
Cumulative waits:	Indicates the percentage of time during the last measurement period wait events of this type occurred.	Percent	Compare the value of this measure across wait types to know which type of waits have occurred frequently during the last measurement period.

3.6.13 SQL Database Log Test

This test monitors the usage of the database logs on the Microsoft SQL server.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**,

Performance as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for an Microsoft SQL server instance being monitored

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Microsoft SQL log file size:	Indicates the size of all the log files in the database during the last measurement period.	KB	An unusually high value may indicate a sudden increase in the size of the log file of the SQL server.
Microsoft SQL log file usage:	Indicates the percentage of log file space that is in use.	Percent	

3.6.14 SQL Index Fragmentation Test

Fragmentation exists when indexes have pages in which the logical ordering, based on the key value, does not match the physical ordering inside the data file. All leaf pages of an index contain pointers to the next and the previous pages in the index. This forms a doubly linked list of all index/data pages. Ideally, the physical order of the pages in the data file should match the logical ordering. Overall disk throughput is increased significantly when the physical ordering matches the logical ordering of the data. This leads to much better performance for certain types of queries. When the physical ordering does not match the logical ordering, disk throughput can become less efficient, because the disk head must move back and forth to gather the index pages instead of scanning forward in one direction. This is how fragmentation affects I/O performance.

The first step to resolving the performance threat posed by fragmented indexes is to identify which indexes are fragmented. The **SQL Index Fragmentation** test helps in this regard. This test scans the indexes on an Microsoft SQL server for high and very high levels of fragmentation, and reports the count of fragmented indexes. Using the detailed diagnosis capability of the test, you can also quickly drill down to the specific indexes that have been fragmented. You can thus proceed to defragment/rebuild the affected indexes, so as to increase disk throughput and improve overall SQL performance.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every *DisplayName* configured for the **OBJECT NAME** parameter of this test

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be monitored. The default value of this parameter is “default”. To monitor an Microsoft SQL instance named “CFS”, enter this as the value of the **INSTANCE** parameter.
6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the ‘SQL server and Windows’ authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if ‘Windows only’ authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **OBJECT NAME** - Specify a comma-separated list of tables, the indexes of which need to be checked

for fragmentation. Every table name should be specified in the following format: *<DisplayName>:<schema_name>.<table_name>*, where *schema_name* refers to the name of the table owner, and *table_name* refers to the name of the table. The *DisplayName* in your specification will appear as the descriptor of this test. For instance, to monitor the indexes of the alarm and history tables owned by user admin, your specification would be: *AlarmMon1:admin.alarm,AlarmMon2:admin.history*. To monitor all tables in aschema, the specification would be of the following format: *<DisplayName>:<schema_name>.**. For example, to monitor all the tables in the admin schema, your specification would be: *AlarmMon:admin.**.

11. **QUERYTIMEOUT** - Specify the time period upto which a query has to wait to obtain the required result set from the database in the **QUERYTIMEOUT** text box. If the query is not successful or if the query waits for a time period exceeding the specified time limit, the test will automatically kill the query.
12. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
13. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
14. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation
Highly fragmented SQL indexes:	Indicates the number of highly fragmented indexes.	Number	If 30% - 49% of an index is found to be fragmented, then such an index is counted as a highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates high index

Measurement	Description	Measurement Unit	Interpretation
			fragmentation. High levels of fragmentation can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify highly fragmented indexes. Once the affected indexes are isolated, take the necessary steps to correct the fragmentation. Towards this end, SQL provides the following statements: a. DROP INDEX followed by CREATE INDEX b. CREATE INDEX WITH DROP_EXISTING c. DBCC INDEXDEFRAG d. DBCC DBREINDEX
Very highly fragmented SQL indexes:	Indicates the number of indexes that are very highly fragmented.	Number	If over 50% of an index is found to be fragmented, then such an index is counted as a highly fragmented index. Ideally, the value of this measure should be 0. A high value indicates high index fragmentation. High levels of fragmentation can cause disk I/O to mount, queries to run for long periods, and the overall performance of the database server to deteriorate. Use the detailed diagnosis of this measure to identify highly fragmented indexes. Once the affected indexes are isolated, take the necessary steps to correct the fragmentation. Towards this end, SQL provides the following statements:

Measurement	Description	Measurement Unit	Interpretation
			a. DROP INDEX followed by CREATE INDEX b. CREATE INDEX WITH DROP_EXISTING c. DBCC INDEXDEFRAG d. DBCC DBREINDEX

3.6.15 SQL Table Size Test

When faced with a disk space crunch on their critical SQL servers, administrators may want to know which databases are hogging the disk space, and which tables on each database have grown beyond permissible limits. The **SQL Table Size** test provides administrators with this information. The test auto-discovers the databases with tables that exceed a configured size limit, and reports the count of such 'large sized tables' for each database. You can use the detailed diagnosis of the test to know which tables in a database are of a large size.

This test has been disabled by default. To enable this test, go to the **ENABLE / DISABLE TESTS** page using the menu sequence : Agents -> Tests -> Enable/Disable, pick *Microsoft SQL* as the **Component type**, *Performance* as the **Test type**, choose this test from the **DISABLED TESTS** list, and click on the **<<** button to move the test to the **ENABLED TESTS** list. Finally, click the **Update** button.

Note:

This test is applicable only to Microsoft SQL Server 2005 (and above).

Target of the test : A Microsoft SQL server

Agent deploying the test : An internal agent

Outputs of the test : One set of results for every database containing tables that are of size greater than the configured **TABLE SIZE GB**

Configurable parameters for the test

1. **TEST PERIOD** - How often should the test be executed
2. **HOST** – The IP address of the Microsoft SQL server.
3. **PORT** - The port number through which the Microsoft SQL server communicates. The default port is 1433.
4. **SSL** – If the Microsoft SQL server being monitored is an SSL-enabled server, then set the **SSL** flag to **Yes**. If not, then set the **SSL** flag to **No**.
5. **INSTANCE** - In this text box, enter the name of a specific Microsoft SQL instance that is to be

monitored. The default value of this parameter is "default". To monitor an Microsoft SQL instance named "CFS", enter this as the value of the **INSTANCE** parameter.

6. **USER** – If a Microsoft SQL Server 7.0/2000 is monitored, then provide the name of a SQL userwith the **Sysadmin** role in this text box. While monitoring a Microsoft SQL Server 2005/2008/2012, provide the name of a SQL user with the **CONNECT SQL, VIEW SERVER STATE, VIEW ANY DEFINITION, VIEW ANY DATABASE**, and **PUBLIC ROLES** in this text box.
7. **PASSWORD** - The password of the specified **USER**
8. **CONFIRM PASSWORD** - Confirm the password by retyping it.
9. **DOMAIN** - By default, *none* is displayed in the **DOMAIN** text box. If the 'SQL server and Windows' authentication has been enabled for the server being monitored, then the **DOMAIN** can continue to be *none*. On the other hand, if 'Windows only' authentication has been enabled, then, in the **DOMAIN** text box, specify the Windows domain in which the managed Microsoft SQL server exists. Also, in such a case, the **USER** name and **PASSWORD** that you provide should be that of a user authorized to access the monitored SQL server.
10. **TABLE SIZE GB** - The test will report the count of those tables that are of a size greater than the value (in GB) specified here.
11. **ISNTLMV2** - In some Windows networks, *NTLM* (*NT LAN Manager*) may be enabled. NTLM is a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users. NTLM version 2 ("NTLMv2") was concocted to address the security issues present in NTLM. By default, the **ISNTLMV2** flag is set to **No**, indicating that NTLMv2 is not enabled by default on the target Microsoft SQL host. Set this flag to **Yes** if NTLMv2 is enabled on the target host.
12. **ISPASSIVE** – If the value chosen is **YES**, then the Microsoft SQL server under consideration is a passive server in a SQL cluster. No alerts will be generated if the server is not running. Measures will be reported as "Not applicable' by the agent if the server is not up.
13. **DETAILED DIAGNOSIS** – To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.

    The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:

    - The eG manager license should allow the detailed diagnosis capability
    - Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0.

Measurements made by the test

Measurement	Description	Measurement Unit	Interpretation	
Table Count:	Indicates the number of tables in this database that are currently of a size greater than the value (in GB) configured against table size gb.	Number	Use the detailed diagnosis of this measure to know which tables in a database are consuming the maximum disk space.	

3.7 The MS SQL Application Dashboard

In order to ascertain how well an application is/has been performing, analysis of the performance of the **System** and **Network** layers of that application alone might not suffice. A closer look at the health of the **Application Layers** is also necessary, so as to promptly detect instantaneous operational issues with the target application, and also proactively identify persistent problems or a consistent performance degradation experienced by the application. To provide administrators with such in-depth insights into overall application performance and to enable them to accurately isolate the root-cause of any application-level slowdown, eG Enterprise offers the **Application Dashboard**. Each of the critical applications monitored by eG Enterprise is accompanied by an exclusive application dashboard. The contents of the dashboard will therefore primarily vary depending upon the application being monitored. Figure 3.25 for instance depicts the **Application Dashboard** of a **MS SQL application**.

Figure 3.25: The Application Dashboard of a MS SQL application

In addition, like the **System** and **Network** dashboards, the contents of the **Application** dashboard too are further governed by the **Subsystem** chosen from Figure 3.25. By default, the **Overview** option is chosen from the **Subsystem** list. If need be, you can change this default setting by picking a different option from the **Subsystem** list. The sections that follow will discuss each of the **Subsystems** offered by the sample **MS SQL application dashboard** shown in Figure 3.25 above.

3.7.1 Overview

The **Overview** dashboard of a MS SQL application provides an all-round view of the health of the MS SQL application being monitored, and helps administrators pinpoint the problem areas. Using this dashboard therefore, you can determine the following quickly and easily:

- Has the application encountered any issue currently? If so, what is the issue and how critical is it?
- How problem-prone has the application been during the last 24 hours? Which application layer has been badly hit?
- Has the administrative staff been able to resolve all past issues? On an average, how long do the administrative personnel take to resolve an issue?
- Are all the key performance parameters of the application operating normally?
- What is the Application configuration of the MS SQL application?
- How many SQL Processes are running?
- What is the Database Usage? What is the Data space and how much Unused space is available in the MS SQL application with respect to each Database?
- How effective is the Database Transaction? What is the Transaction rate of each Database? Are the transactions for each Database behaving normally or is there any abnormal transactional behavior that has been reported during a particular time period?

The contents of the **Overview Dashboard** have been elaborated on hereunder:

1. The **Current Application Alerts** section of Figure 3.25 reveals the number and type of issues currently affecting the performance of the MS SQL application that is being monitored. To know more about the current issues, click on any cell against **Distribution** that represents the problem priority of interest to you; the details of the current problems of that priority will then appear as depicted by Figure 3.26.

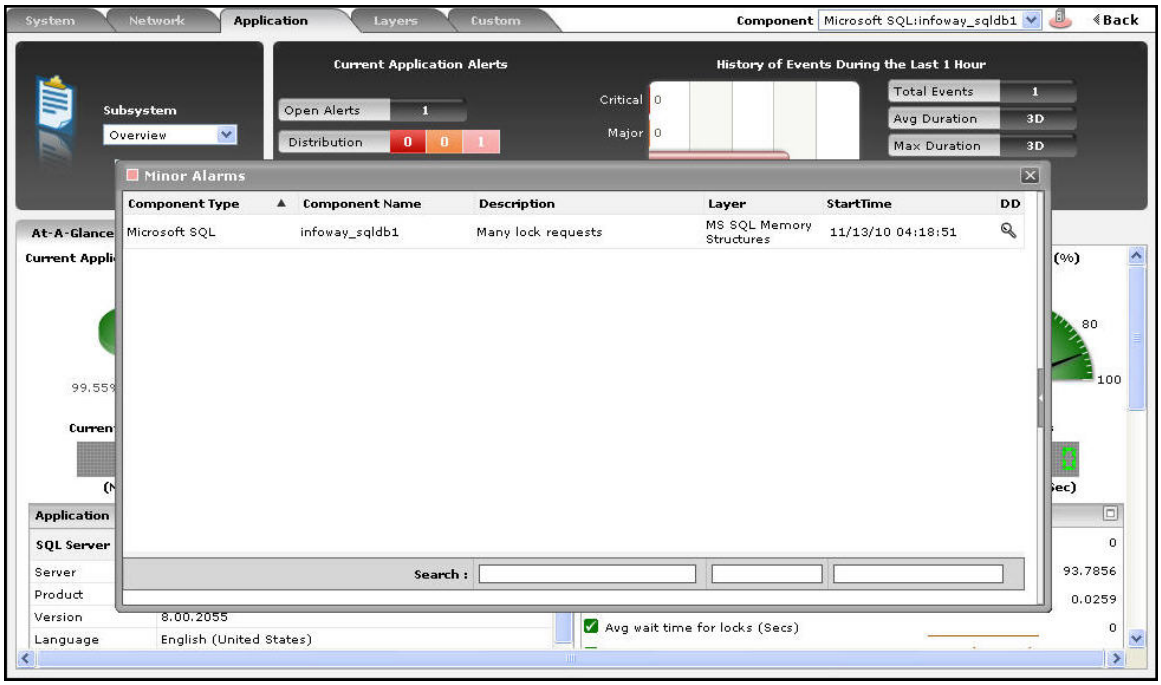


Figure 3.26: Viewing the current application alerts of a particular priority

2. If the pop-up window of Figure 3.26 reveals too many problems, you can use the **Search** text boxes that have been provided at the end of the **Description**, **Layer**, and **StartTime** columns to run quick searches on the contents of these columns, so that the alarm of your interest can be easily located. For instance, to find the alarm with a specific description, you can provide the whole/part of the alarm description in the text box at the end of the **Description** column in Figure 3.26; this will result in the automatic display of all the alarms with descriptions that contain the specified search string.
3. To zoom into the exact layer, test, and measure that reported any of the listed problems, click on a particular alarm in the **Alarms** window of Figure 3.26. Doing so will introduce an **Alarm Details** section into the **Alarms** window (see Figure 3.27), which provides the complete information related to the problem clicked on. These details include the **Site** affected by the problem for which the alarm was raised, the test that reported the problem, and the last measure that was reported will be reported in the **Last Measures**.

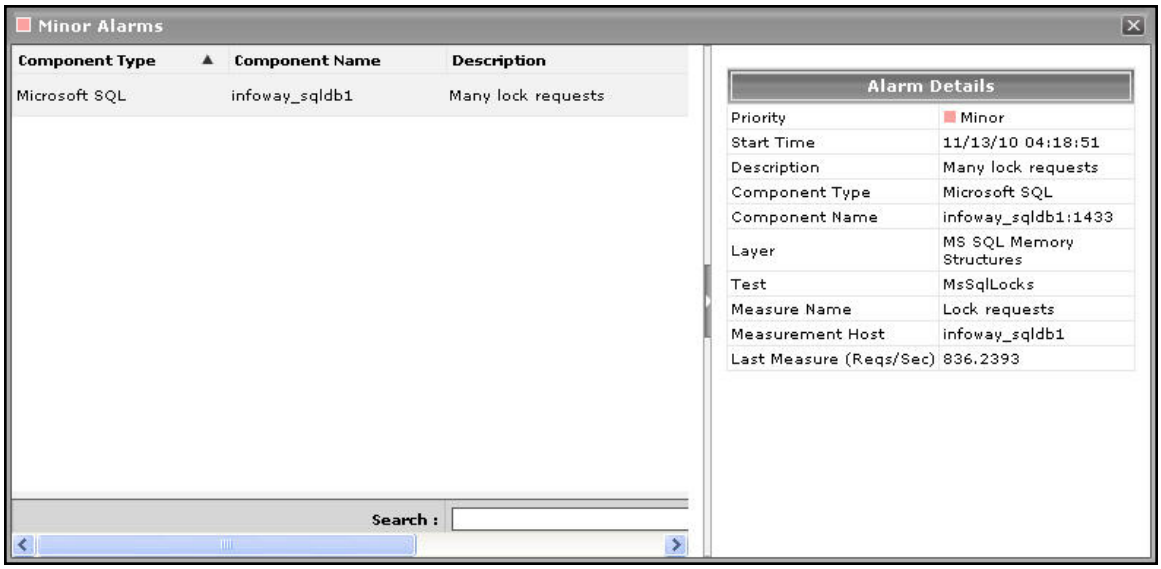


Figure 3.27: Additional alarm details

4. While the list of current issues faced by the application serves as a good indicator of the current state of the application, to know how healthy/otherwise the application has been over time, a look at the problem history of the application is essential. Therefore, the dashboard provides the **History of Events** section; this section presents a bar chart, where every bar indicates the number of problems of a particular severity, which was experienced by the MS SQL application during the last 1 hour (by default). Clicking on a bar here will lead you to Figure 3.28 which provides a detailed history of problems of that priority. Alongside the bar chart, you will also find a table displaying the average and maximum duration for problem resolution; this table helps you determine the efficiency of your administrative staff.

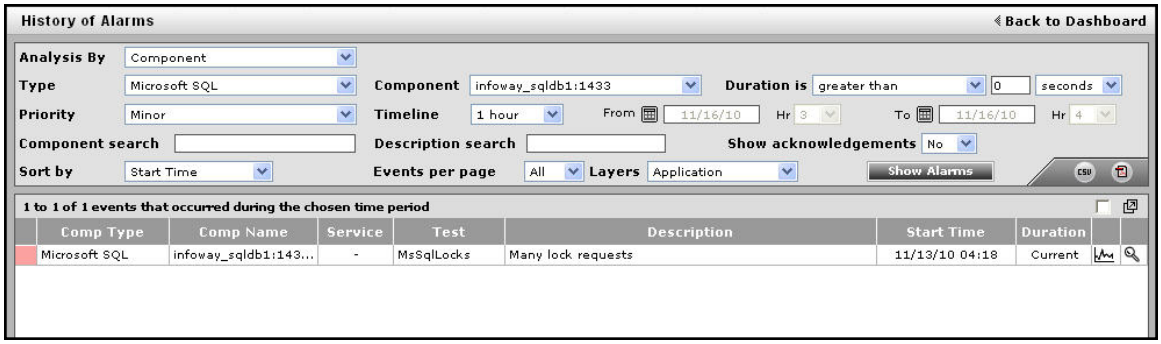


Figure 3.28: The problem history of the target application

5. If required, you can override the default time period of 1 hour of the event history, by following the steps below:

   - Click the  button at the top of the dashboard to invoke the **Dashboard Settings** window.
   - Select the **Event History** option from the **Default timeline for** list.
   - Set a different default timeline by selecting an option from the **Timeline** list.
   - Finally, click the **Update** button.

6. Back in the dashboard, you will find that the **History of Events** section is followed by an **At-A-Glance** section; this section, using pie charts, digital displays and gauge charts, reveals, at a single glance, the current status of some of the critical metrics and key components of the MS SQL application. For instance, the **Current Application Health** pie chart indicates the current health of the application by representing the number of application-related metrics that are in various states. Clicking on a slice here will take you to Figure 3.28 that provides a detailed problem history.
7. The dial and digital graphs that follow provide you with quick updates on the status of a pre-configured set of resource usage-related metrics pertaining to the MS SQL application. If required, you can configure the dial graphs to display the threshold values of the corresponding measures along with their actual values, so that deviations can be easily detected. For this purpose, do the following:
   - Click the  button at the top of the dashboard to invoke the **Dashboard Settings** window.
   - Set the **Show Thresholds** flag in the window to **Yes**.
   - Finally, click the **Update** button.

   You can customize the **At-A-Glance** tab page further by overriding the default measure list for which dial/digital graphs are being displayed in that tab. To achieve this, do the following:

   - Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
   - To add measures for the dial graph, pick the **Dial Graph** option from the **Add/Delete Measures for** list. Upon selection of the **Dial Graph** option, the pre-configured measures for the dial graph will appear in the **Existing Value(s)** list. Similarly, to add a measure to the digital display, pick the **Digital Graph** option from the **Add/Delete Measures for** list. In this case, the **Existing Value(s)** list box will display all those measures for which digital displays pre-exist.
   - Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list. **Note that while configuring measures for a dial graph the 'Measures' list will display only those measures that report percentage values**.

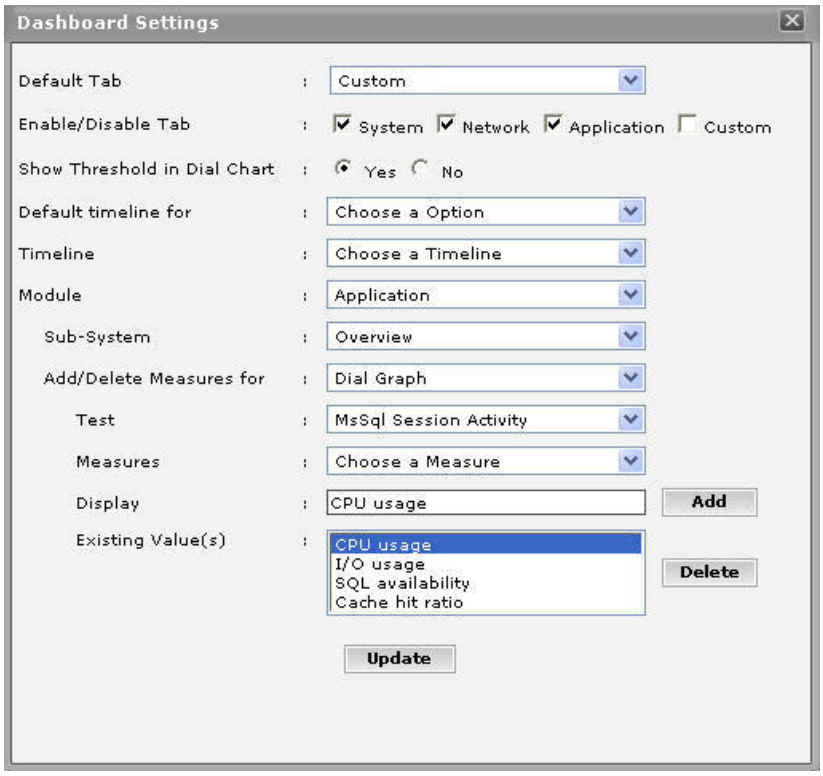


Figure 3.29: Configuring measures for the dial graph

- If you want to delete one/more measures from the dial/digital graphs, then, as soon as you choose the **Dial Graph** or **Digital Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

- Clicking on a dial/digital graph will lead you to the layer model page of the MS SQL application; this page will display the exact layer-test combination that reports the measure represented by the

dial/digital graph.

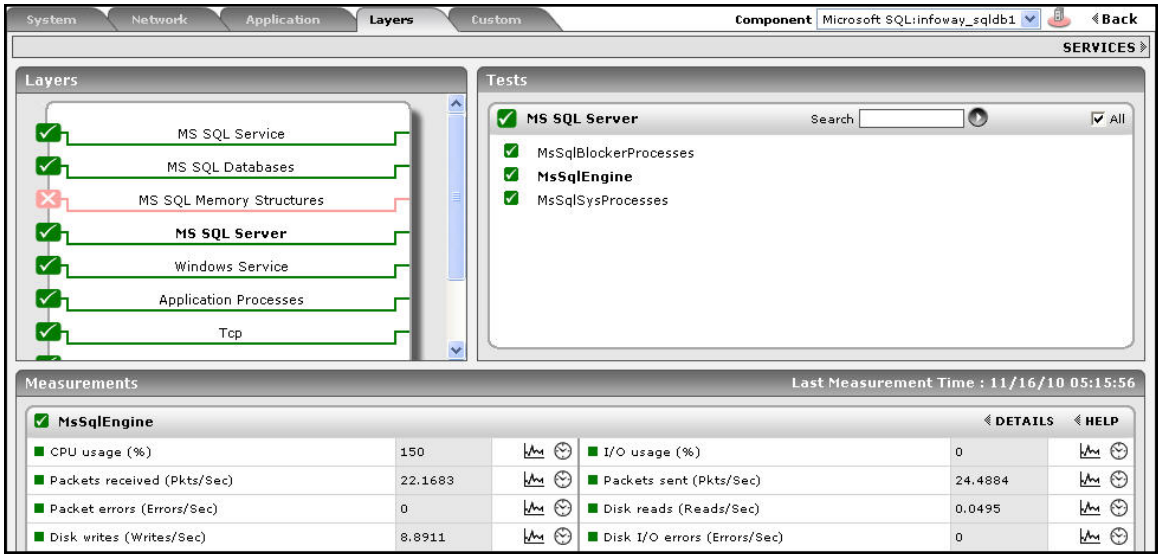


Figure 3.30: The page that appears when the dial/digital graph in the Overview dashboard of the MS SQL Application is clicked

8. If your eG license enables the **Configuration Management** capability, then, an **Application Configuration** section will appear here (as shown in Figure 3.25) providing the basic configuration of the application. You can configure the type of configuration data that is to be displayed in this section by following the steps below:

   - Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
   - To add more configuration information to this section, first, pick the **Application Configuration** option from the **Add/Delete Measures for** list. Upon selection of this option, all the configuration measures that pre-exist in the **Configuration Management** section will appear in the **Existing Value(s)** list.
   - Next, select the config **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
   - If you want to delete one/more measures from this section, then, as soon as you choose the **Application Configuration** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
   - Finally, click the **Update** button to register the changes.

9. Next to this section, you will find a pre-configured list of **Key Performance Indicators** of the MS SQL application. Besides indicating the current state of and current values reported by a default set of resource usage metrics, this section also reveals 'miniature' graphs of each measure, so that you can instantly study how that measure has behaved during the last 1 hour (by default) and thus determine whether the change in state of the measure was triggered by a sudden dip in performance or a consistent one. Clicking on a measure here will lead you to Figure 3.31, which displays the layer and test that reports the measure.

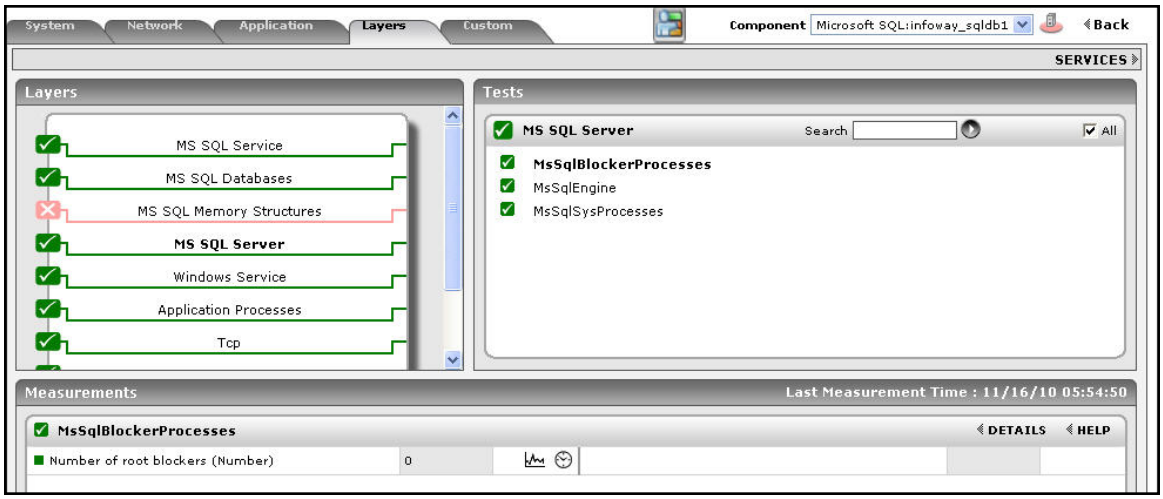


Figure 3.31: Clicking on a Key Performance Indicator

You can, if required, override the default measure list in the **Key Performance Indicators** section by adding more critical measures to the list or by removing one/more existing ones from the list. For this, do the following:

- Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
- To add more metrics to the **Key Performance Indicators** section, first, pick the **Performance Indicator** option from the **Add/Delete Measures for** list. Upon selection of this option, all the measures that pre-exist in the **Key Performance Indicators** section will appear in the **Existing Value (s)** list.
- Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
- If you want to delete one/more measures from this section, then, as soon as you choose the **Key Performance Indicators** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

10. Clicking on a 'miniature' graph that corresponds to a key performance indicator will enlarge the graph (see Figure 3.32), so that you can view and analyze the measure behavior more clearly, and can also alter the **Timeline** and dimension (**3D**/ **2D**) of the graph, if need be.

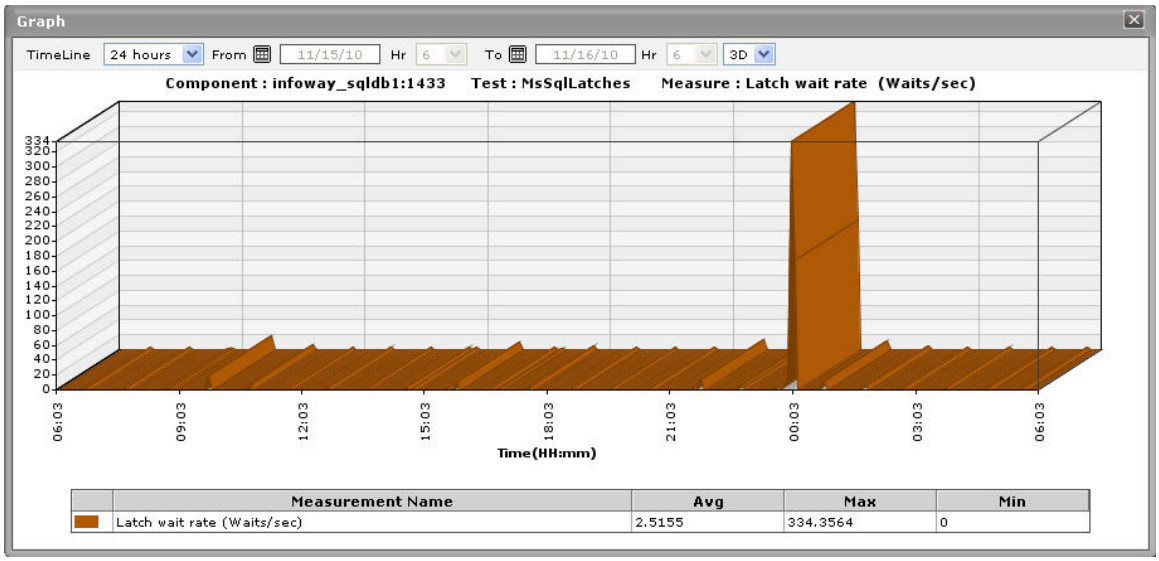


Figure 3.32: Enlarging the Key Performance Indicator graph

11. This way, the first few sections of the **At-A-Glance** tab page help understand what issues are currently affecting the application health, and when they actually originated. To diagnose the root-cause of these issues however, you would have to take help from the remaining sections of the **At-A-Glance** tab page. For instance, the **Key Performance Indicators** section may indicate a sudden/steady increase in the Log cache hit ratio of the MS SQL application. However, to determine whether the rise in the Log cache hit ratio was a result of one/more high SQL processes executing on the MS SQL application or a couple of resource-intensive SQL applications, you need to focus on the **SQL Process - Summary** section. This **SQL Process - Summary** section for starters reveals the number of Processes that are in varying states of activity. With the help of this section therefore, you can quickly figure out whether there are currently any:

    - Processes that are being processed in the background;
    - Processes that are being blocked by other processes;
    - Processes that are not utilized at present by the MS SQL application;
    - Processes that are currently running for the target MS SQL application, etc.

    Say, you notice that too many processes are currently running in a BACKGROUND state. Immediately, you might want to know whether this is a sudden occurrence, or has that problem occurred over a course of time. To enable you to determine this, every process that is displayed in the **SQL Process - Summary** section is accompanied by a 'miniature' graph, which tracks the changes in the corresponding process during the last 1 hour (by default). To enlarge the graph, click on it; this will invoke Figure 3.33. The enlarged graph allows you to change the **Timeline** for analysis, and also the graph dimension.

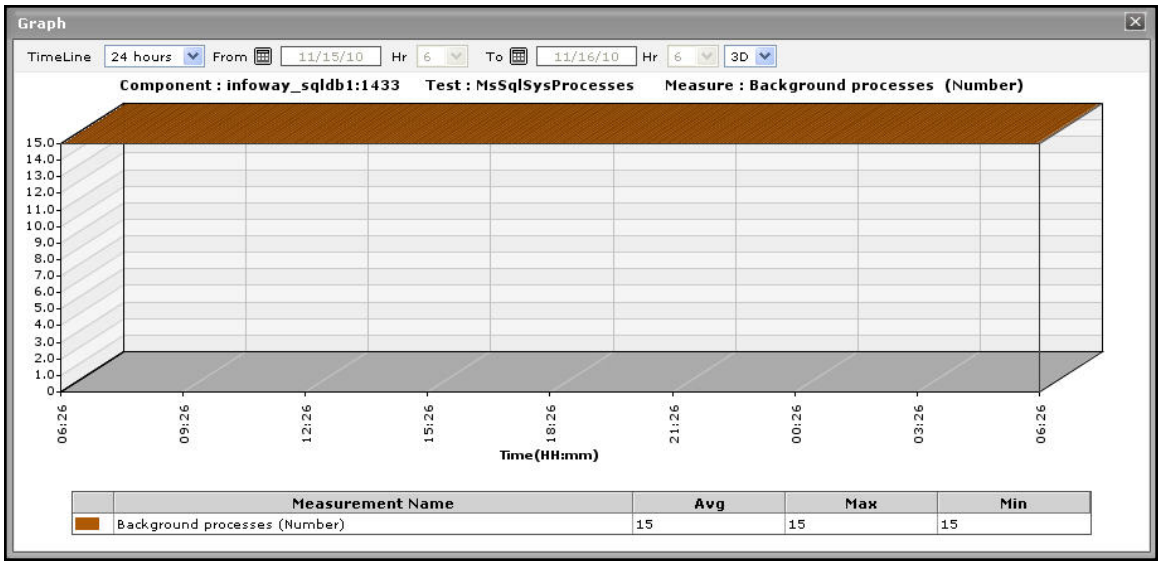


Figure 3.33: The enlarged processes graph

12. The **Database Usage Summary** section reveals how well the databases are being managed by the target MS SQL application. For every database, this section reveals the space that is utilized by the data files, the space that is used for indexing the data, and the space that is currently unused in the database. From this information, you can infer which database is utilizing the maximum amount of allocated resources. By default, the Database list provided by this section is sorted in the alphabetical order of the names of the databases. If need be, you can change the sort order so that the databases are arranged in, say, the descending order of values displayed in the **Data Space** column - this column displays the space that is utilized by the data files. To achieve this, simply click on the column heading **Data Space**. Doing so tags the **Data Space** label with a **down arrow** icon - this icon indicates that the **Database Usage Summary** table is currently sorted in the descending order of the space used by the data files. To change the sort order to 'ascending', all you need to do is just click again on the **Data Space** label or the **down arrow** icon. Similarly, you can sort the table based on any column available in it.

13. The **Database Transaction Summary** section, on the other hand, provides the transaction details of each of the databases that are currently available in the MS SQL application. By default, the database list provided by this section is sorted in the alphabetical order of the process names. If need be, you can change the sort order so that the databases are arranged in, say, the descending order of values displayed in the **Active transactions** column - this column displays the number of active transactions made by each database that is available in the MS SQL application. To achieve this, simply click on the column heading – **Active transactions**. Doing so tags the **Active transactions** label with a **down arrow** icon - this icon indicates that the database list is currently sorted in the descending order of the active transaction count. To change the sort order to 'ascending', all you need to do is just click again on the **Active transactions** label or the **down arrow** icon. Similarly, you can sort the process list based on any column available in the **Database Transaction Summary** section.

14. While the **At-A-Glance** tab page reveals the current state of the databases and the overall resource usage of the MS SQL application, to perform additional diagnosis on problem conditions highlighted by the **At-A-Glance** tab page and to accurately pinpoint their root-cause, you need to switch to the **Details** tab page (see Figure 3.34) by clicking on it. For instance, the **At-A-Glance** tab page may indicate the number of

processes that are currently blocked, but to know which process has been blocked for the longest time, you will have to use the **Details** tab page.

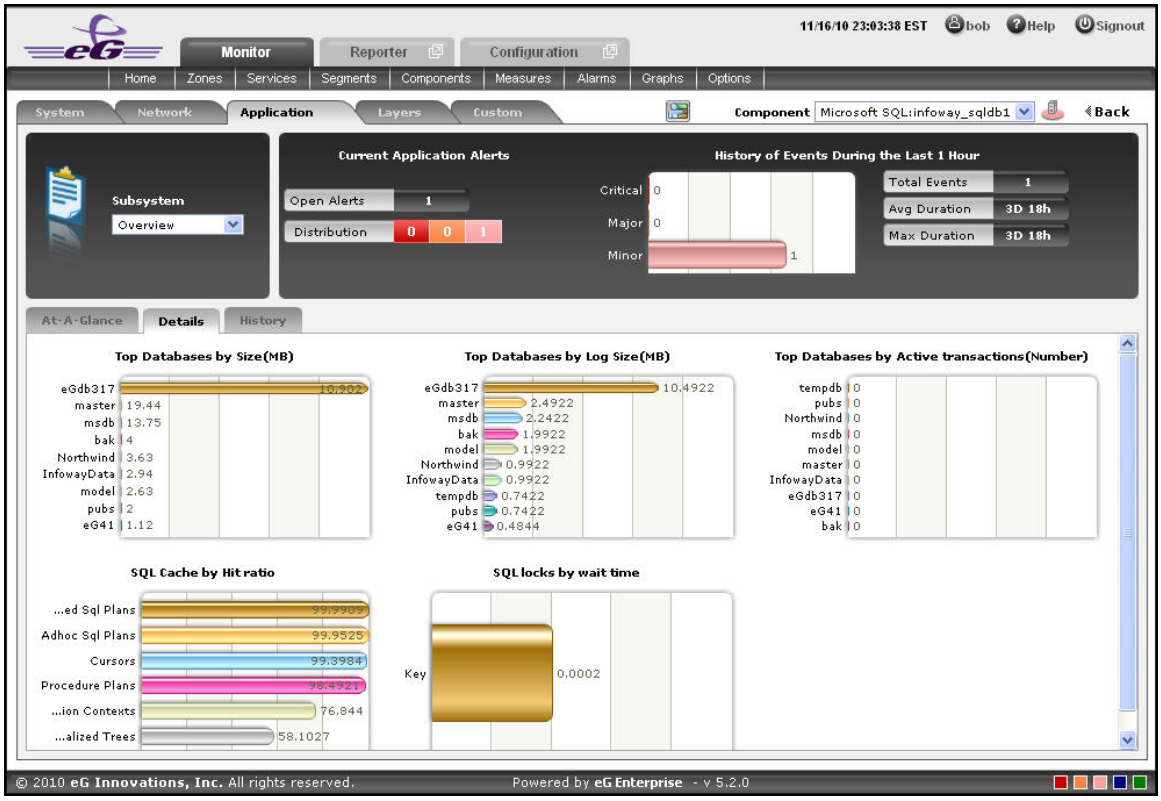


Figure 3.34: The Details tab page of the MS SQL Application Overview Dashboard

15. The **Details** tab page comprises of a default set of comparison bar graphs using which you can accurately determine the following:

- The size of the databases.
- The Log size of the databases.
- How many Active transactions are made on each database?
- What is the SQL Cache hit ratio of each database?

If required, you can configure the **Details** tab page to include comparison graphs for more measures, or can even remove one/more existing graphs by removing the corresponding measures. To achieve this, do the following:

- Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **Overview** from the **Sub-System** list.
- To add measures for comparison graphs, first, pick the **Comparison Graph** option from the **Add/Delete Measures for** list. Upon selection of this option, the pre-configured measures for comparison graphs will appear in the **Existing Value(s)** list.
- Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures**

list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.

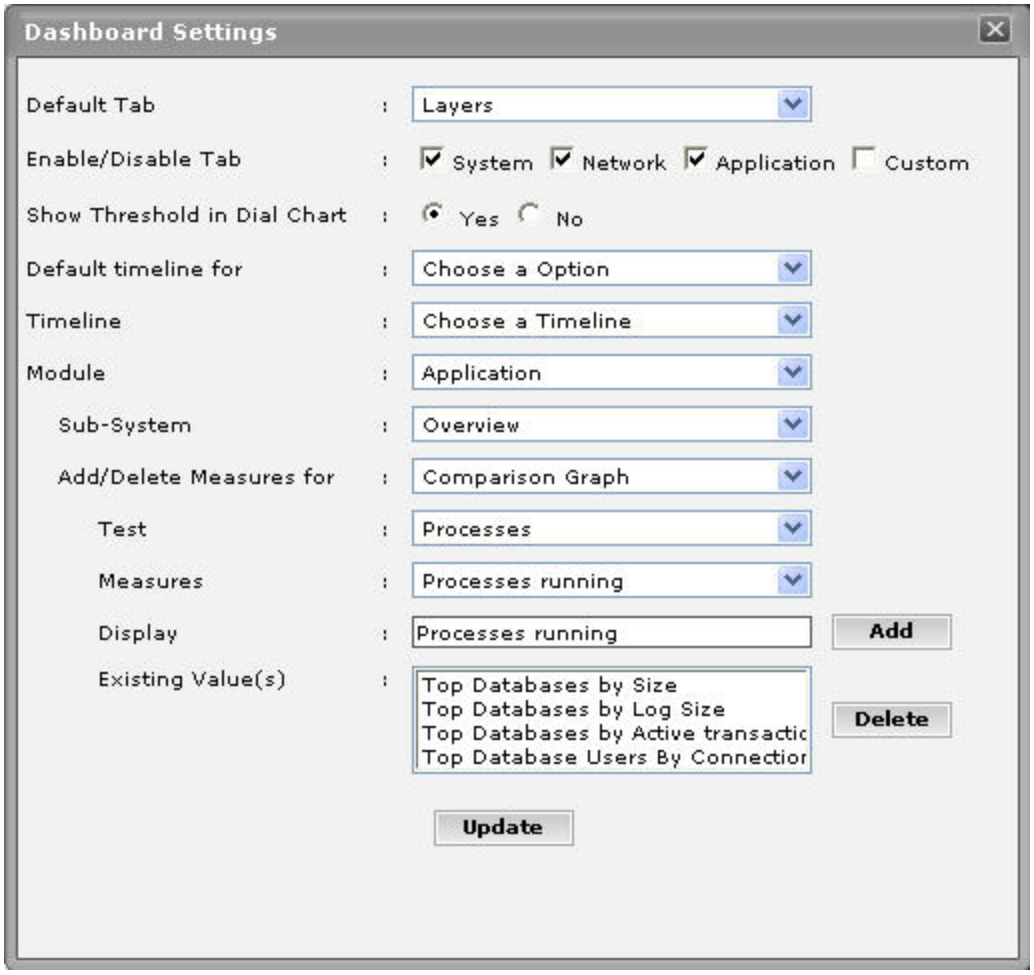


Figure 3.35: Configuring measures for the dial graph

- If you want to delete one/more measures for which comparison graphs pre-exist in the **details** tab page, then, as soon as you choose the **Comparison Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
- Finally, click the **Update** button to register the changes.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

16. By default, the comparison bar graphs list the top-10 databases only. To view the complete list of databases, simply click on the corresponding graph in Figure 3.34. This enlarges the graph as depicted by Figure 3.36.

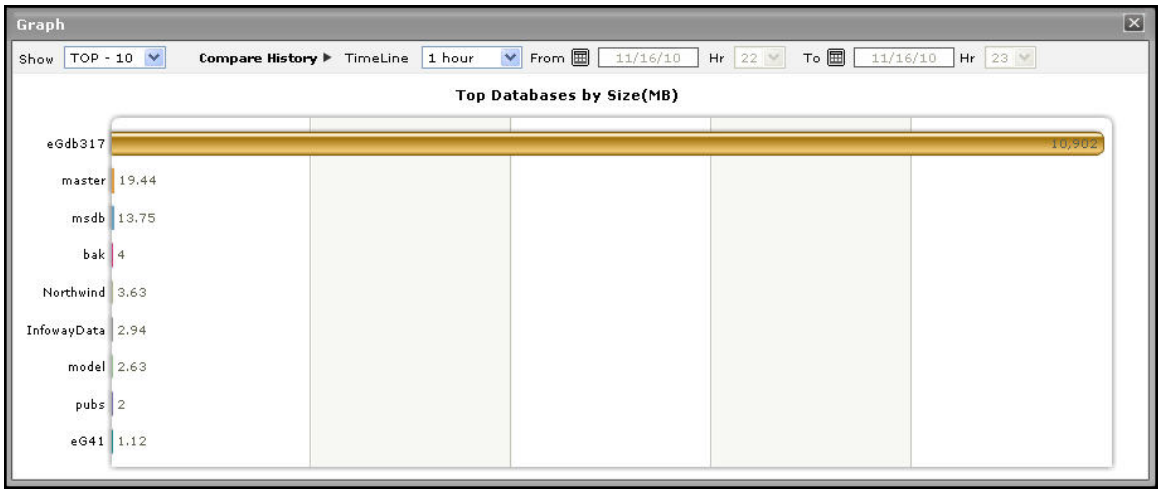
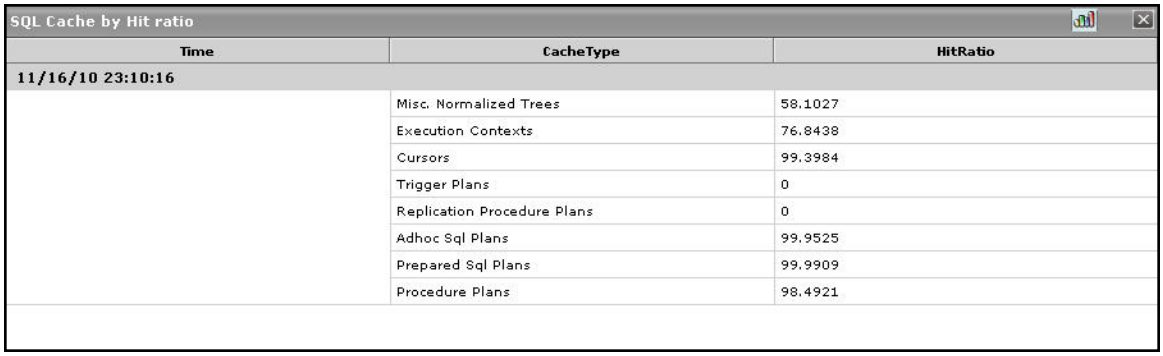


Figure 3.36: The expanded top-n graph in the Details tab page of the MS SQL Application Overview Dashboard

17. Though the enlarged graph lists all the databases in this case by default, you can customize the enlarged graph to display the details of only a few of the larger/smaller databases by picking a **TOP-N** or **LAST-N** option from the **Show** list in Figure 3.36.

18. Another default aspect of the enlarged graph is that it pertains to the current period only. Sometimes however, you might want to know what occurred during a point of time in the past; for instance, while trying to understand the reason behind a sudden increase or decrease in the size of the database usage on a particular day last week, you might want to first determine the database whose size has abnormally increased / decreased on the same day. To figure this out, the enlarged graph allows you to compare the historical performance of the databases. For this purpose, click on the **Compare History** link in Figure 3.36 and select the **TimeLine** of your choice.

19. Where detailed diagnosis is applicable, you can quickly view the detailed measures that correspond to a comparison graph by clicking on the icon at the right, top corner of the enlarged graph. This will invoke Figure 3.37, using which you can arrive at the root-cause of a problem.



Time	CacheType	HitRatio
11/16/10 23:10:16		
	Misc. Normalized Trees	58.1027
	Execution Contexts	76.8438
	Cursors	99.3984
	Trigger Plans	0
	Replication Procedure Plans	0
	Adhoc Sql Plans	99.9525
	Prepared Sql Plans	99.9909
	Procedure Plans	98.4921

Figure 3.37: The detailed diagnosis that appears when the DD icon in the enlarged comparison bar graph is clicked

20. For detailed time-of-day / trend analysis of the historical performance of a MS SQL application, use the **History** tab page. By default, this tab page (see Figure 3.38) provides time-of-day graphs of critical

measures extracted from the target MS SQL application, using which you can understand how performance has varied during the default period of 24 hours. In the event of a problem, these graphs will help you determine whether the problem occurred suddenly or grew with time. To alter the timeline of all the graphs simultaneously, click on the **Timeline** link at the right, top corner of the **History** tab page of Figure 3.38.

You can even override the default timeline (of 24 hours) of the measure graphs, by following the steps below:

- Click on the  icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

Figure 3.38: Time-of-day measure graphs displayed in the History tab page of the Application Overview Dashboard

21. You can click on any of the graphs to enlarge it, and can change the **Timeline** of that graph in the enlarged mode as shown in Figure 3.39.

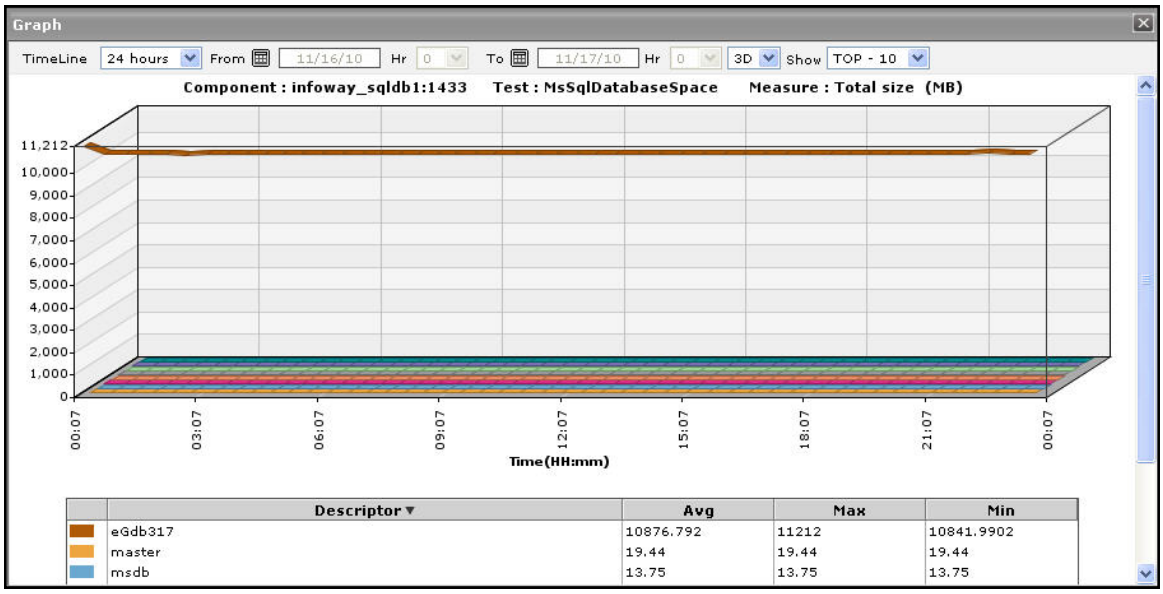


Figure 3.39: An enlarged measure graph of a MS SQL Application

22. In case of tests that support descriptors, the enlarged graph will, by default, plot the values for the **TOP-10** descriptors alone. To configure the graph to plot the values of more or less number of descriptors, select a different **TOP-N / LAST-N** option from the **Show** list in Figure 3.39.

23. If you want to quickly perform service level audits on the MS SQL application, then summary graphs may be more appropriate than the default measure graphs. For instance, a summary graph might come in handy if you want to determine the variation of Total size of a database with respect to the percentage of time during the last 24 hours. Using such a graph, you can determine whether the database size has been constant or varied, and if not, how frequently the application faltered in this regard. To invoke such summary graphs, click on the  icon at the right, top corner of the **History** tab page. Figure 3.40 will then appear.

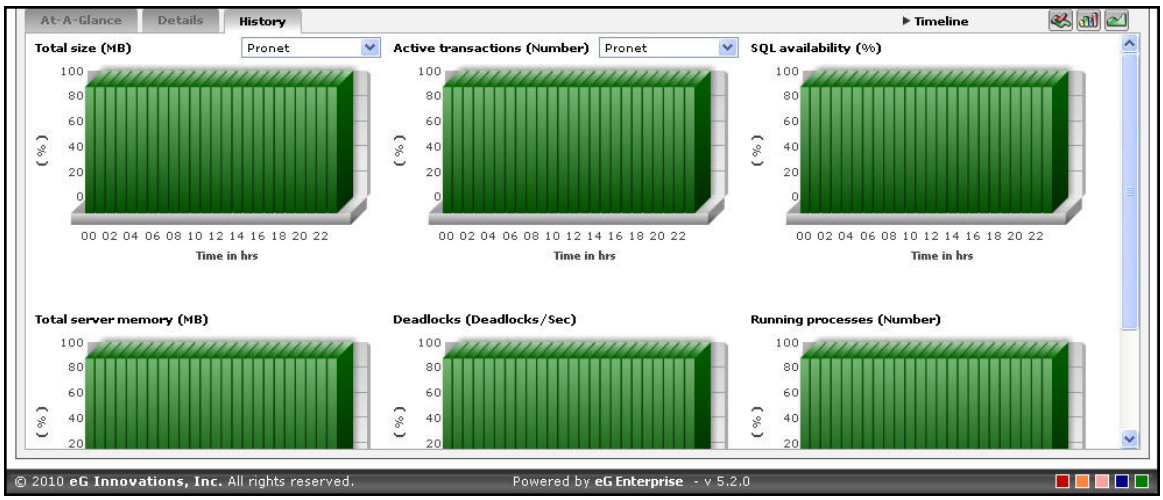


Figure 3.40: Summary graphs displayed in the History tab page of the Application Overview Dashboard

24. You can alter the timeline of all the summary graphs at one shot by clicking the **Timeline** link at the right, top corner of the **History** tab page of Figure 3.38. You can even alter the default timeline (of 24 hours) for these graphs, by following the steps given below:
    - Click on the icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.
25. To change the timeline of a particular graph, click on it; this will enlarge the graph as depicted by Figure 3.41. In the enlarged mode, you can alter the **Timeline** of the graph. Also, though the graph plots hourly summary values by default, you can pick a different **Duration** for the graph in the enlarged mode, so that daily/monthly performance summaries can be analyzed.

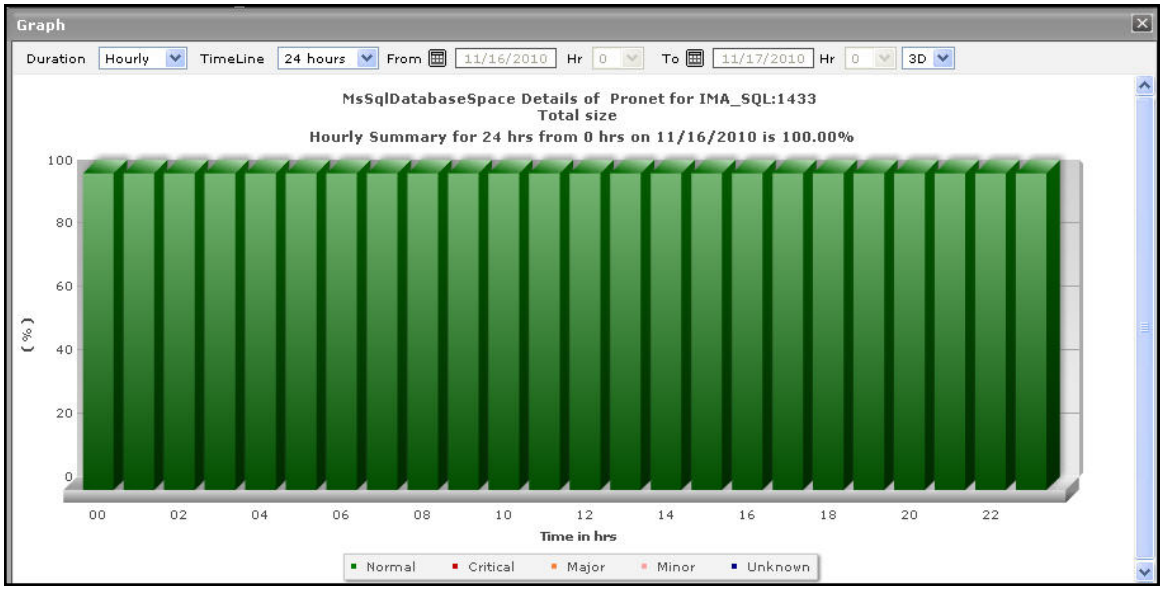


Figure 3.41: An enlarged summary graph of the MS SQL Application

26. To perform effective analysis of the past trends in performance, and to accurately predict future measure behavior, click on the icon at the right, top corner of the **History** tab page. These trend graphs as shown in Figure 3.42 typically show how well and how badly a measure has performed every hour during the last 24 hours (by default). For instance, the Total size trend graph of each database of a MS SQL application will help you figure out the total size of the database that was available in the application every hour during the last 24 hours. If the gap between the minimum and maximum values is marginal, you can conclude that the size of the database has been more or less constant during the designated period; this implies that the size of the database has neither increased nor decreased steeply during the said timeline. On the other hand, a wide gap between the maximum and minimum values is indicative of an erratic change in the size of the database, and may necessitate further investigation.

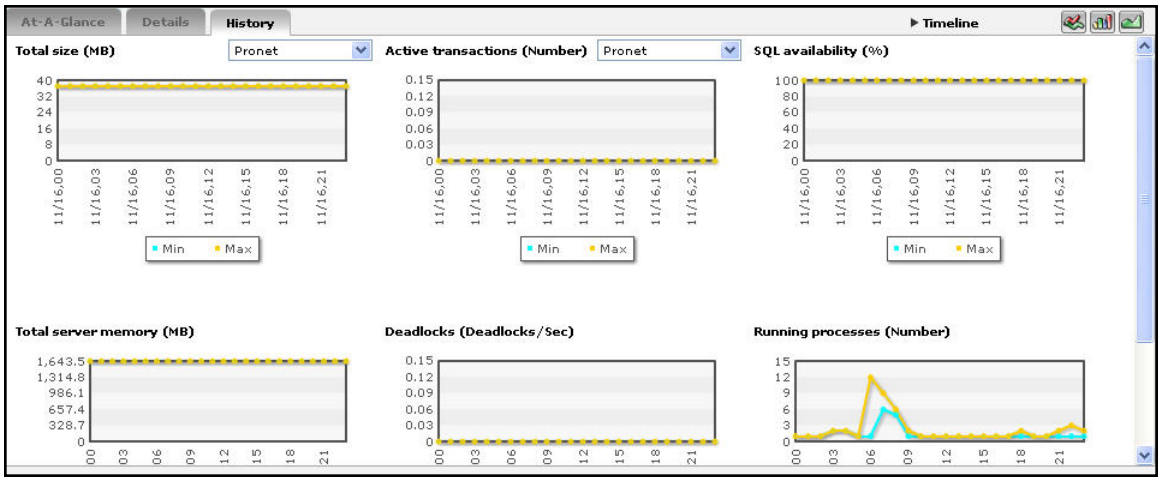


Figure 3.42: Trend graphs displayed in the History tab page of the Application Overview Dashboard

27. To analyze trends over a broader time scale, click on the **Timeline** link at the right, top corner of the **History** tab page, and edit the **Timeline** of the trend graphs. Clicking on any of the miniature graphs in this tab page will enlarge that graph, so that you can view the plotted data more clearly and even change its **Timeline**.

    To override the default timeline (of 24 hours) of the trend graphs, do the following:

    - Click on the  icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.

28. Besides the timeline, you can even change the **Duration** of the trend graph in the enlarged mode. By default, **Hourly** trends are plotted in the trend graph. By picking a different option from the **Duration** list, you can ensure that **Daily** or **Monthly** trends are plotted in the graph instead.

29. Also, by default, the trend graph only plots the minimum and maximum values registered by a measure. Accordingly, the **Graph** type is set to **Min/Max** in the enlarged mode. If need be, you can change the **Graph** type to **Avg** (see Figure 3.43), so that the average trend values of a measure are plotted for the given **Timeline**. For instance, if an average trend graph is plotted for the *Total size* measure, then the resulting graph will enable administrators to ascertain whether the size of a particular database has been constant during a specified timeline.

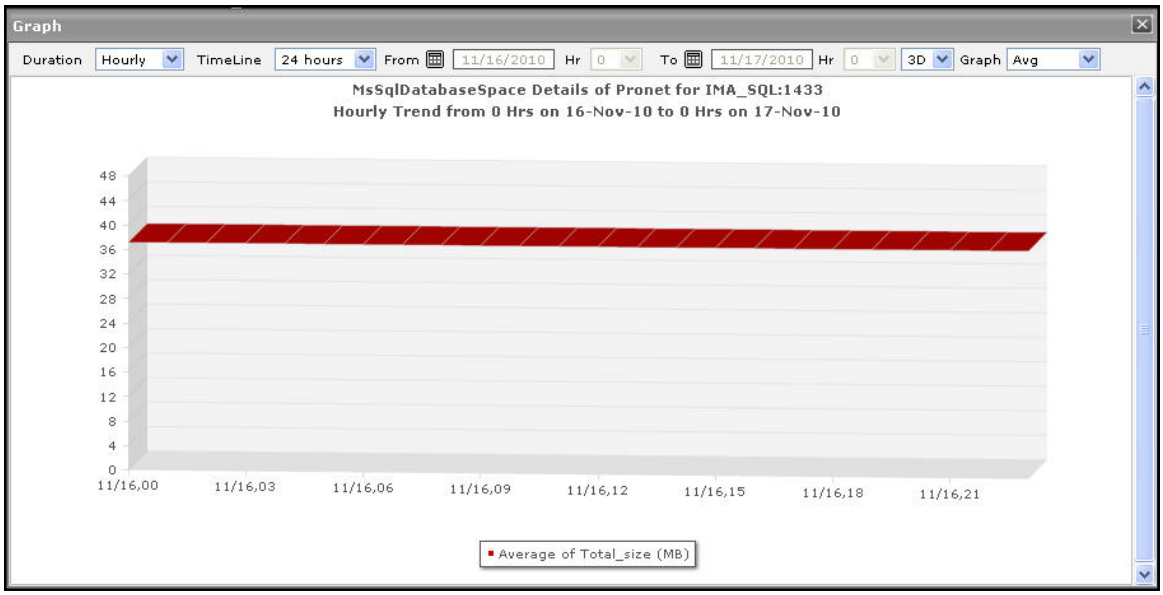


Figure 3.43: Viewing a trend graph that plots average values of a measure for a database available in the MS SQL application

30. Likewise, you can also choose **Sum** as the **Graph** type to view a trend graph that plots the sum of the values of a chosen measure for a specified timeline. For instance, if you plot a 'sum of trends' graph for the measure that reports the Total size of a database available in the MS SQL application, then, the resulting graph will enable you to analyze, on an hourly/daily/monthly basis (depending upon the **Duration** chosen), whether there was any change in the size of the database.

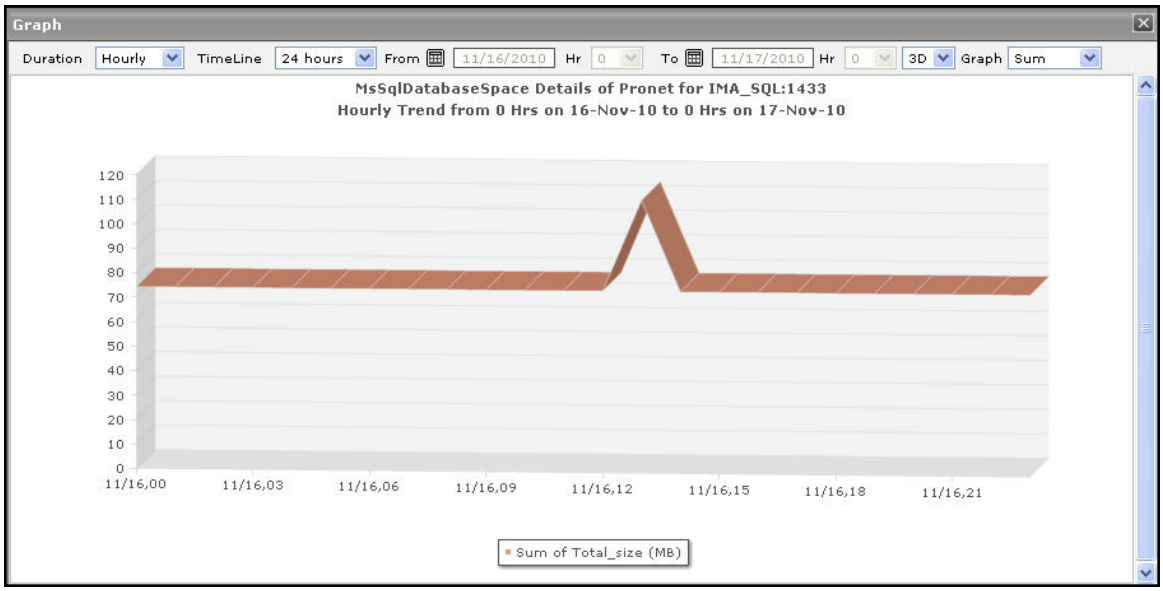


Figure 3.44: A trend graph plotting sum of trends for a database available in the MS SQL application

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

31. At any point in time, you can switch to the measure graphs by clicking on the button.
32. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
    - Click the button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **Overview** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.

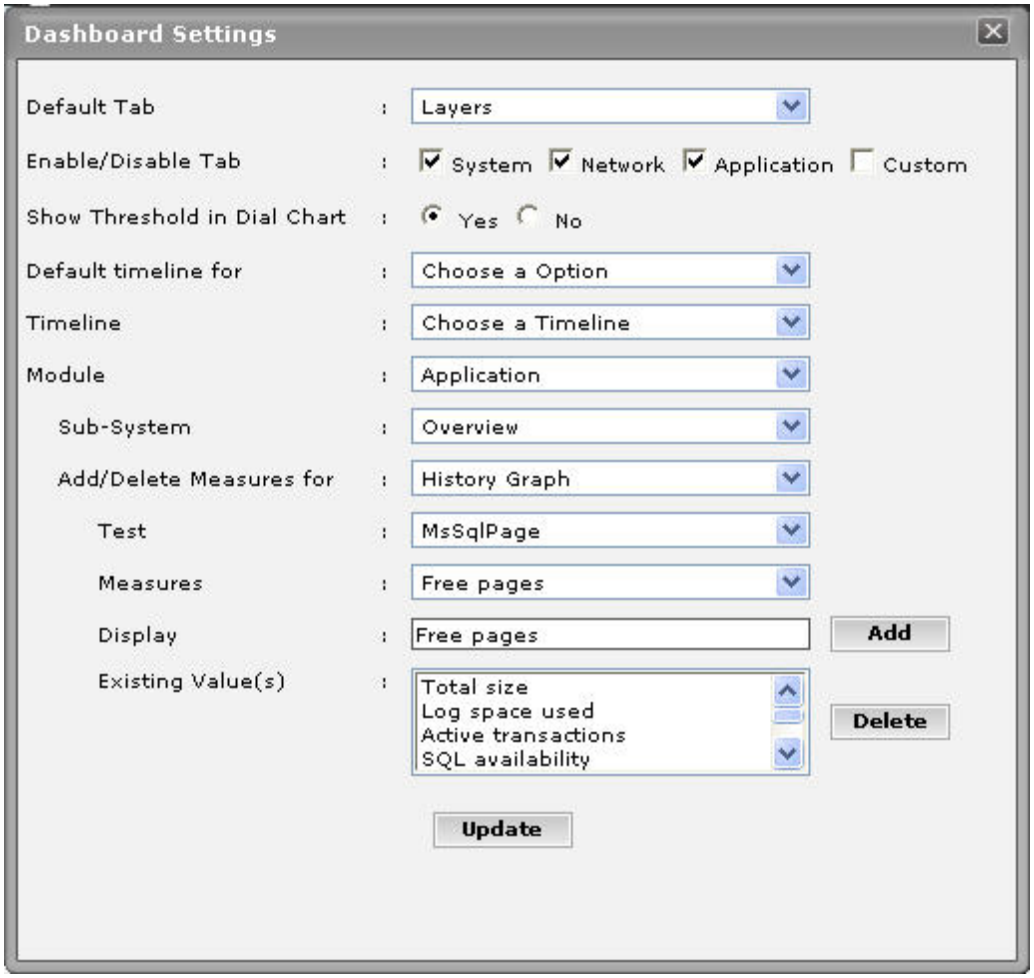


Figure 3.45: Adding a new graph to the **History** tab page

- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.

- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure, to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.2 SQLServer

If you want to assess how efficiently the SQL server manages the databases available on it, and thus promptly detect the server related discrepancies, select the **SQLServer** option from the **Subsystem** list.

Figure 3.46: The SQLServer Dashboard

The contents of the **SQLServer** dashboard that then appears (see Figure 3.46) are as follows:

1. The dashboard begins with a dial and digital graphs section, which enables you to visually track the changes that are happening in the measures related to the SQL server of the MS SQL application. For instance, the CPU usage of the SQL server can be viewed at a single glance. Clicking on a dial/digital graph will lead you to the layer model page of the MS SQL Application; this page will display the exact layer-test combination that reports the measure represented by the dial/digital graph.
2. The **History** tab page displays measure graphs that depict how the server related measures such as CPU usage has been varying over time. In the event of any retaliation in the measures, this time-bound analysis will help you to easily differentiate between a sudden spike in the CPU usage and a consistent rise in the same.
3. By default, these historical graphs track the time-of-day variations in memory usage during the last 24 hours. You can override this default timeline by following the steps discussed below:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.
4. To change the timeline of all the measure graphs at one shot, just click on the **Timeline** link at the right, top corner of the **History** tab page. To alter the timeline for a single graph, just click on that graph - this will enlarge the graph. You can change the **Timeline** of the graph in the enlarged mode.

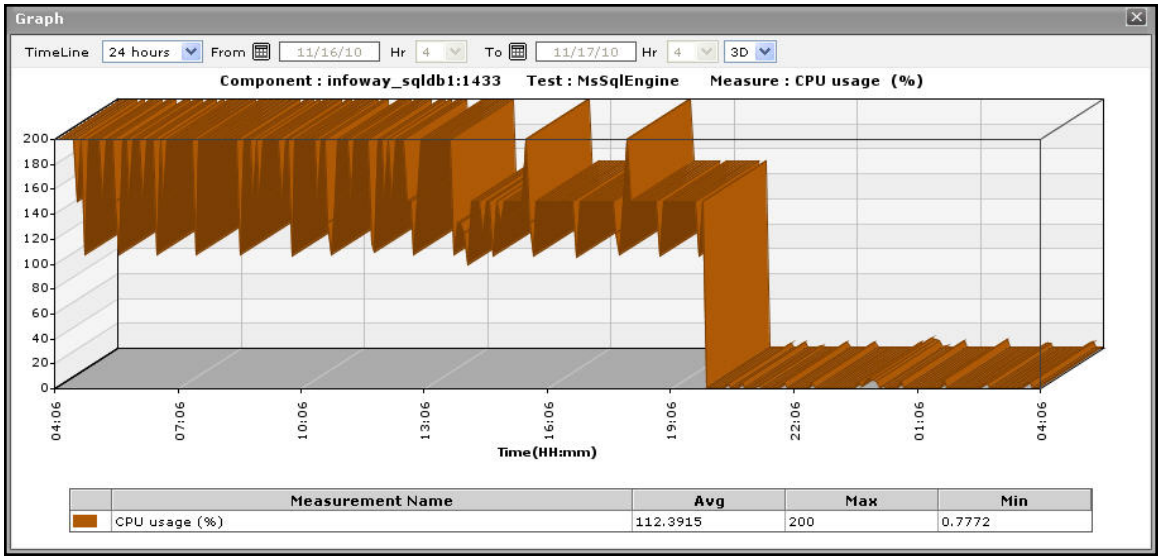


Figure 3.47: An enlarged measure graph in the History tab page of the SQLServer dashboard

5. Instead of these measure graphs, you can, if required, view summary graphs of the memory-related measures in the **History** tab page. For this, click on the  icon at the right, top corner of the **History** tab page. Summary graphs help you figure out the percentage of time during the last 24 hours (by default) the MS SQL application was hogged by the server-related issues. While monitoring mission-critical applications that are governed by rigid service level agreements, summary graphs will help you determine

whether the guaranteed CPU usage levels were fulfilled or not, and if not, how often did the usage levels slip.

6. You can override the default timeline (of 24 hours) of the summary graphs by following the steps discussed below:

   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

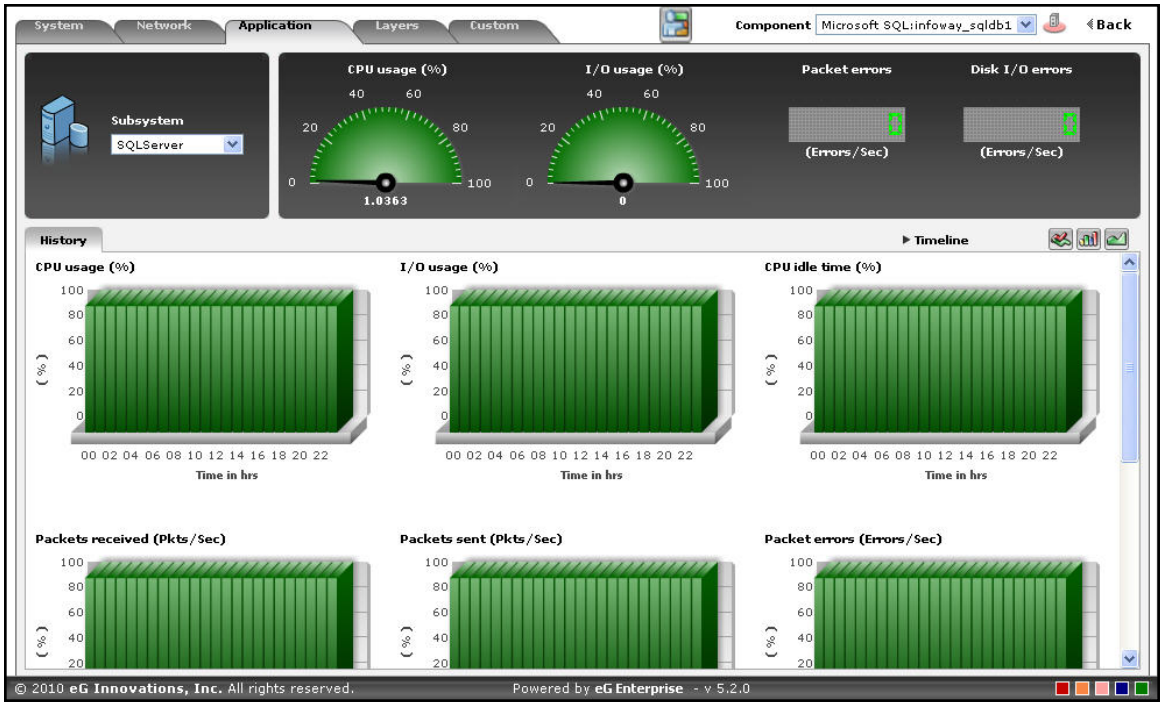


Figure 3.48: Summary graphs displayed in the History tab page of the SQLServer Dashboard

7. Here again, you can change the **Timeline** of all the summary graphs by clicking on the **Timeline** link in Figure 3.48, or click on a graph, enlarge it, and change its **Timeline** in the enlarged mode. Also, though the graph plots hourly summary values by default, you can pick a different **Duration** for the graph in the enlarged mode, so that daily/monthly performance summaries can be analyzed.

8. You can click on the  icon at the right, top corner of the **History** tab page to view trend graphs of the memory usage-related measures. By default, these trend graphs plot the maximum and minimum memory usage values for every hour of the last 24 hours (by default). The default timeline of 24 hours can be overridden by following the steps discussed below:

- Click on the  icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

Using these trend graphs, you can understand the variations in the CPU usage of the SQL server during the last 24 hours (by default), deduce the future usage trends, and accordingly recommend changes to the server size.

Figure 3.49: Trend graphs displayed in the History tab page of the SQLServer Dashboard

9. Here again, you can change the **Timeline** of all the trend graphs by clicking on the **Timeline** link in Figure 3.49, or click on a graph, enlarge it, and change its **Timeline** in the enlarged mode. Also, though the graph plots hourly trend values by default, you can pick a different **Duration** for the graph in the enlarged mode, so that daily/monthly performance trends can be analyzed.
10. Also, by default, the trend graph only plots the minimum and maximum values registered by a measure. Accordingly, the **Graph** type is set to **Min/Max** in the enlarged mode. If need be, you can change the **Graph** type to **Avg**, so that the average trend values of a measure are plotted for the given **Timeline**. Such a graph will enable you to assess whether the memory resources were utilized effectively or not, over time.
11. Likewise, you can also choose **Sum** as the **Graph** type to view a trend graph that plots the sum of the values of a chosen measure for a specified timeline. For instance, a 'sum of trends' CPU usage will enable you to analyze, on an hourly/daily/monthly basis (depending upon the **Duration** chosen), how the CPU

usage of a server has varied during the specified timeline.

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

12. At any point in time, you can switch to the measure graphs by clicking on the  button.
13. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
    - Click the  button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **SQL Server** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
    - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
    - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
    - Next, select the **Measure** of interest.
    - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
    - This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

    Note:

    Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.3 SQLMemory Test

If you want to assess how efficiently the Microsoft SQL application uses the memory resources available to it, and thus promptly detect issues related to the memory-intensive measures, select the **SQLMemory** option from the **Subsystem** list.

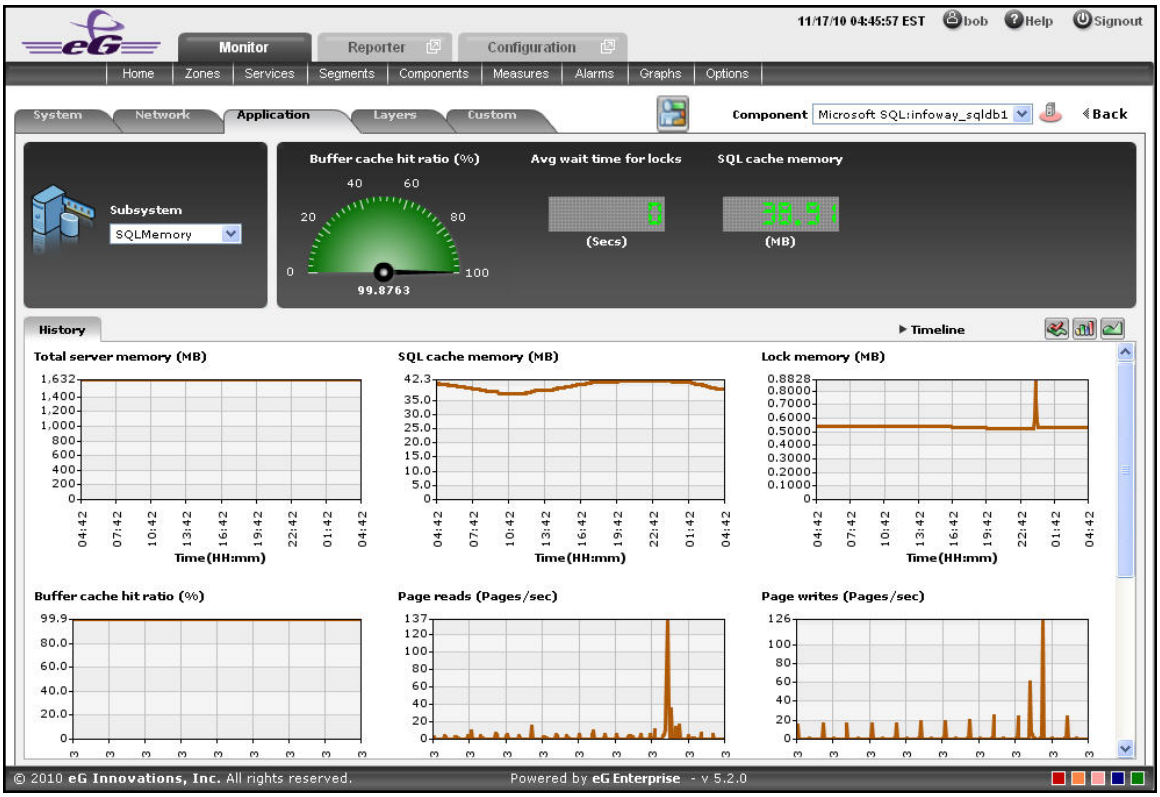


Figure 3.50: Figure 3.49: The SQLMemory Dashboard

The contents of this dashboard are discussed hereunder:

1. The dashboard begins with a dial and digital graphs section, which enables you to visually track the changes that are happening in the measures related to the memory related measures that are available in the Microsoft SQL application. For instance, the Buffer cache hit ratio pertaining to the SQL Memory can be viewed at a single glance. Clicking on a dial/digital graph will lead you to the layer model page of the Microsoft SQL Application; this page will display the exact layer-test combination that reports the measure represented by the dial/digital graph.
2. The **History** tab page displays time-of-day graphs for all the memory-related measures for a default time duration of 24 hours. You can override this default timeline (of 24 hours) by following the steps below:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

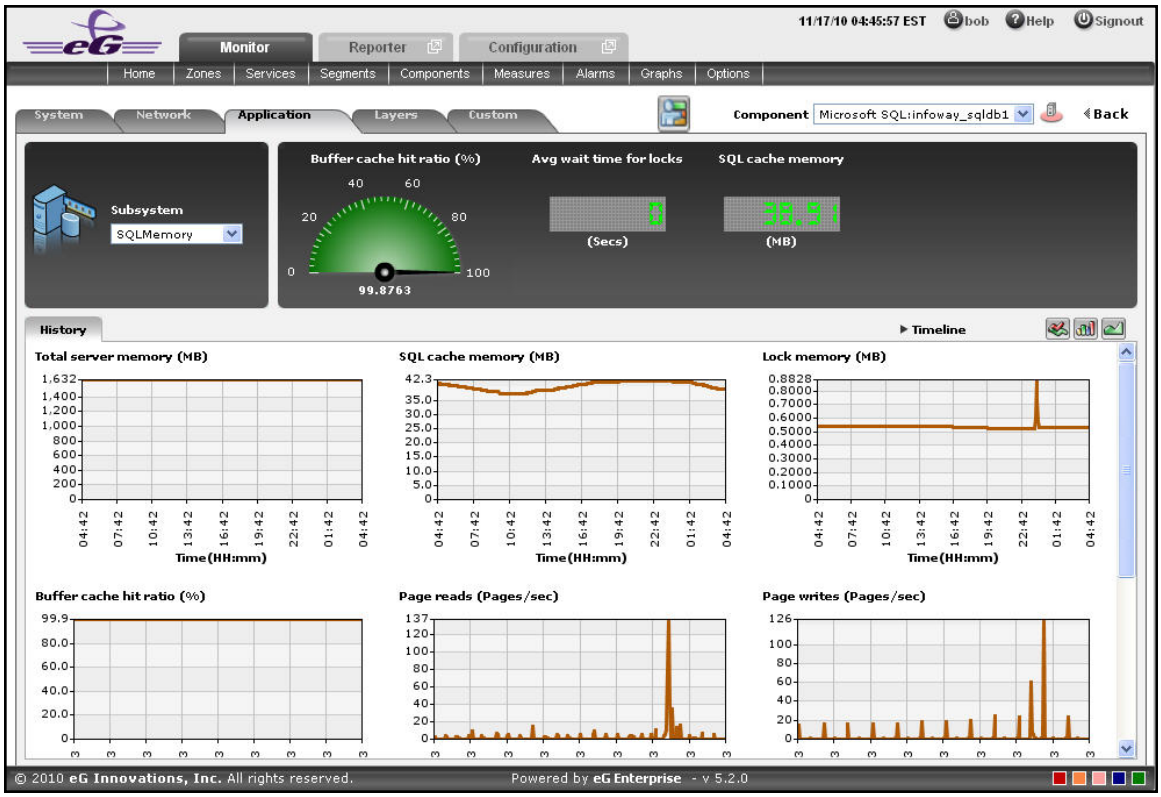


Figure 3.51: The History tab page of the SQLMemory Dashboard

3. Say, you suddenly notice that the Page reads measure has increased; in such a case, you can use these measure graphs to figure out when during the last 24 hours there was an increase in the number of pages that was read per second. If required, you can even look beyond the last 24 hours - i.e., you can find out whether the anomaly originated much earlier. For this, you just need to click on the graph of interest to you. This will enlarge the graph; in the enlarged mode, you can alter the graph **Timeline**, so that the performance of that measure can be analyzed over a broader time window. In this mode, you can even change the graph dimension from **3D** to **2D,** or vice-versa.

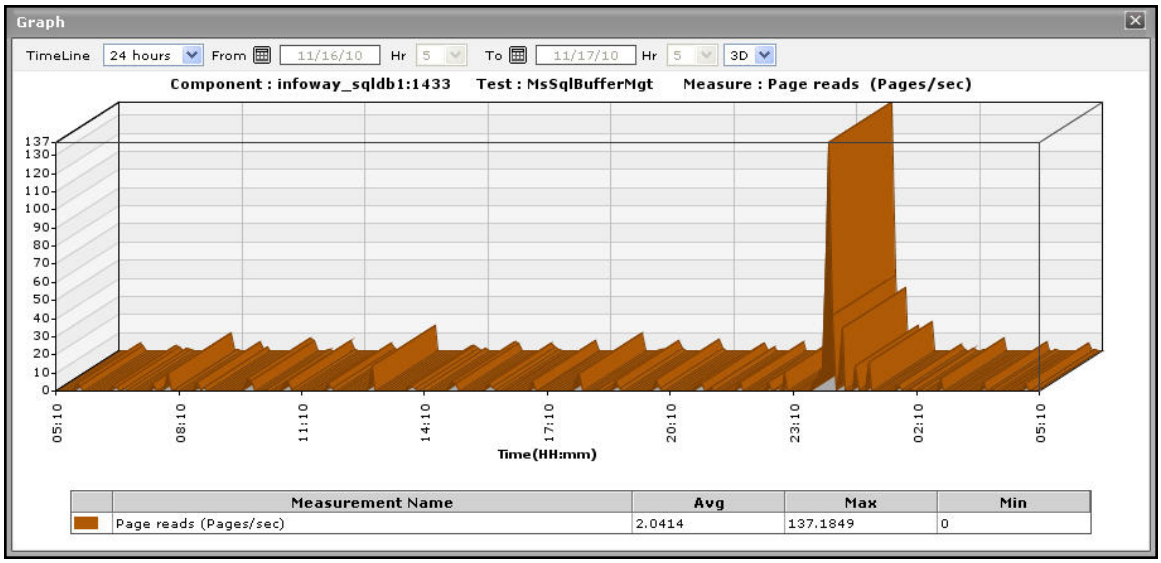


Figure 3.52: An enlarged measure graph in the History tab page of the SQLMemory dashboard

4. To view summary graphs of these memory-related measures instead of the default measure graphs, just click on the icon at the right, top corner of the **History** tab page. Figure 3.53 will then appear. The summary graphs of Figure 3.53 reveal the percentage of time during the last 24 hours (by default) the Microsoft SQL application has been affected by memory-related issues, and the type of issues (whether critical/major/minor) the application was experiencing. These graphs help determine whether the assured service levels were delivered or not.
5. The default duration (of 24 hours) of the summary graphs can be overridden by following the procedure discussed below:
   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

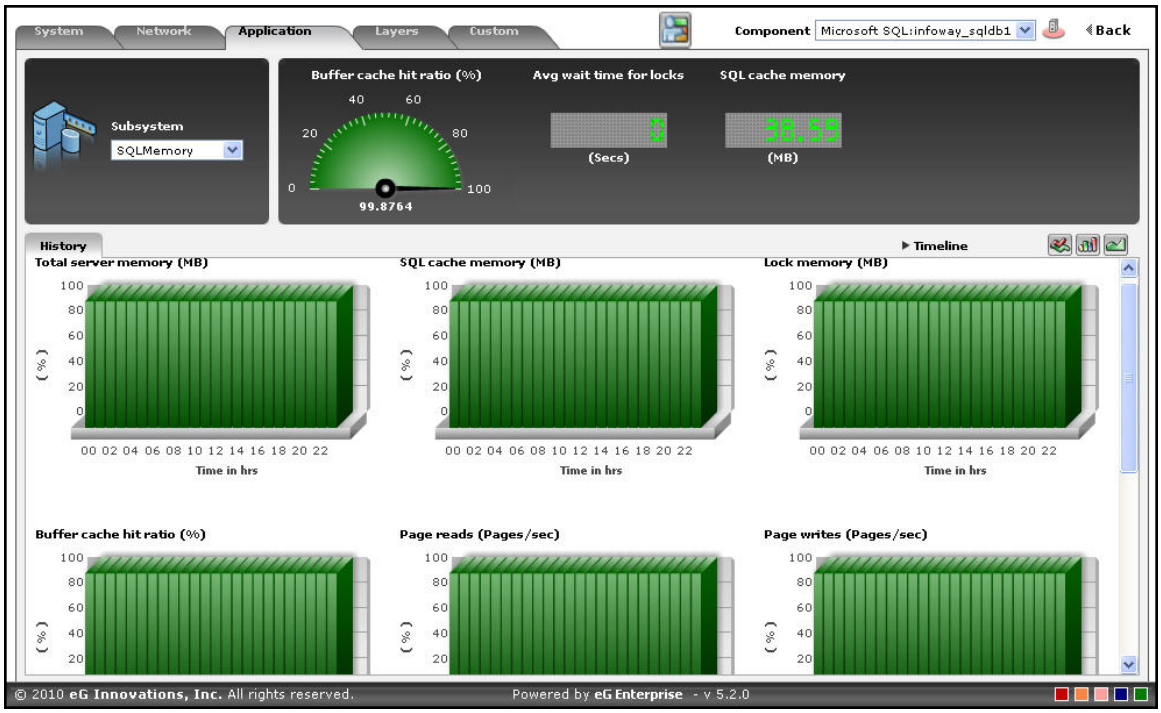


Figure 3.53: Summary graphs displayed in the History tab page of the SQLMemory Dashboard

6. Use the **Timeline** link at the right, top corner of the tab page to change the timeline of all the summary graphs at one shot. For altering the timeline of a single graph, click on it; this will enlarge the graph. In the enlarged mode, you can change the **Timeline** of the summary graph and modify the dimension (3D/2D) of the graph. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode.

7. If you want to view the past trends in the memory performance, click on the icon at the right, top corner of the **History** tab page. Figure 3.54 will then appear. Using the trend graphs displayed in Figure 3.54, you can better assess the current capacity of your application and can accordingly plan its future capacity. By default, these trend graphs plot the maximum and minimum values registered by every memory-related measure during every hour of the last 24 hours. From this data, you can clearly figure out when during the last 24 hours the application performance has peaked and when it has been below-normal.

8. The default duration (of 24 hours) of the trend graphs can be overridden by following the procedure discussed below:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

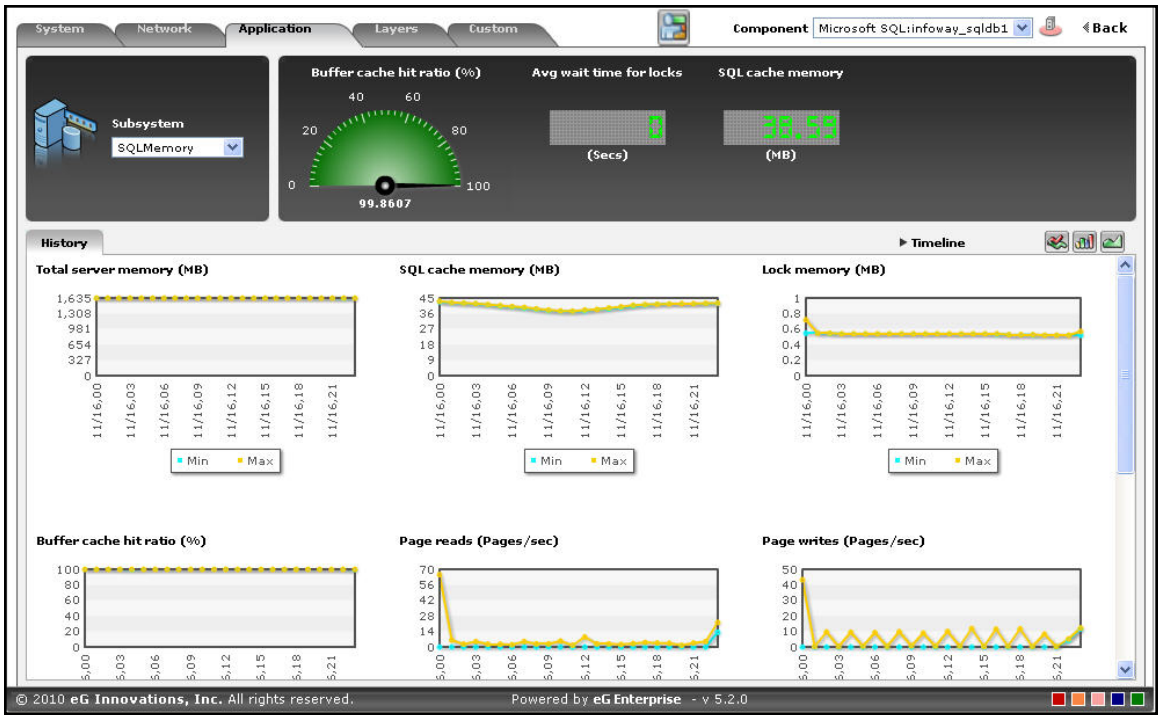


Figure 3.54: Trend graphs displayed in the History tab page of the SQLMemory Dashboard

9. Use the **Timeline** link at the right, top corner of the tab page to change the timeline of all the trend graphs at one shot. For altering the timeline of a single graph, click on it; this will enlarge the graph. In the enlarged mode, you can change the **Timeline** of the trend graph and modify the dimension (3D/2D) of the graph. Also, by default, hourly trends are plotted in the trend graph; you can configure these graphs to plot daily/monthly trend values instead by picking the relevant option from the **Duration** list in the enlarged mode. Moreover, by default, the trend graphs plot only the minimum and maximum values registered by a measure during the specified timeline - this graph will enable you to isolate those times at which performance of that measure had peaked and the times it had fared poorly. If need be, you can select the **Avg** option from the **Graph type** list in the enlarged mode to make sure that the trend graph plots the average trend values for the specified timeline. Alternatively, you can select the **Sum** option from the **Graph type** list to have the trend graph plot the sum of trends for the specified timeline.

   Note:

   In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

10. At any point in time, you can switch to the measure graphs by clicking on the  button.

11. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:

- Click the  button at the top of the dashboard.
- The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **SQLMemory** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.4 SQLProcesses

Select the **SQLProcesses** option from the **Subsystem** list to know how efficiently the class loader used by the Java application is and has been loading/unloading classes onto memory. Upon selection, Figure 3.55 will appear.

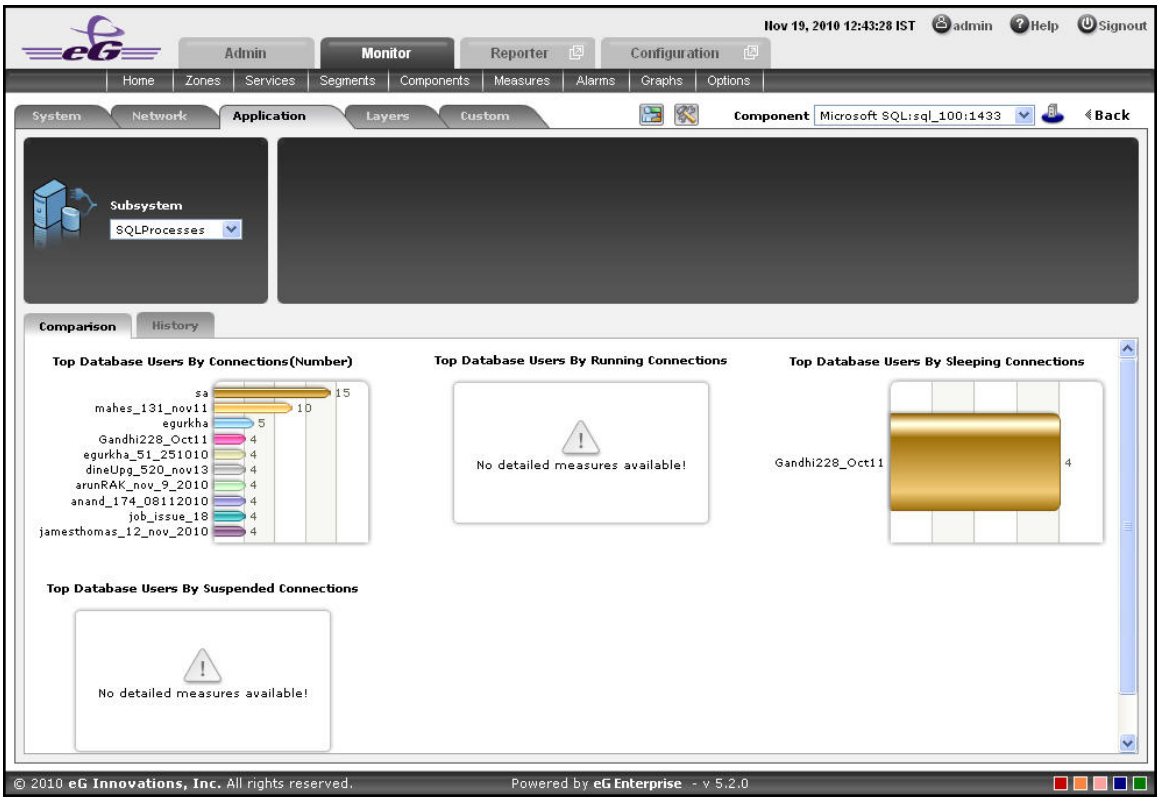


Figure 3.55: The Comparison tab page of the SQLProcesses Dashboard

The contents of this dashboard are as follows:

1. The **Comparison** tab page provides a series of top 10 charts, using which you can isolate the top Database Users for various resource-intensive processes. This default list of measures for top-n chart generation can be overridden by following the steps discussed below:

   - Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **SQLProcesses** from the **Sub-System** list.
   - To add new measures for which top-n graphs are to be displayed in the **Comparison** tab page, first, pick the **Comparison Graph** option from the **Add/Delete Measures for** list. Upon selection of this option, the pre-configured measures for comparison graphs will appear in the **Existing Value(s)** list.
   - Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
   - If you want to delete one/more measures for which comparison graphs pre-exist in the **Comparison** tab page, then, as soon as you choose the **Comparison Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
   - Finally, click the **Update** button to register the changes.

   Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

2. To view the complete list of database users for each process, simply click on the corresponding graph in Figure 3.55. This enlarges the graph as depicted by Figure 3.56.

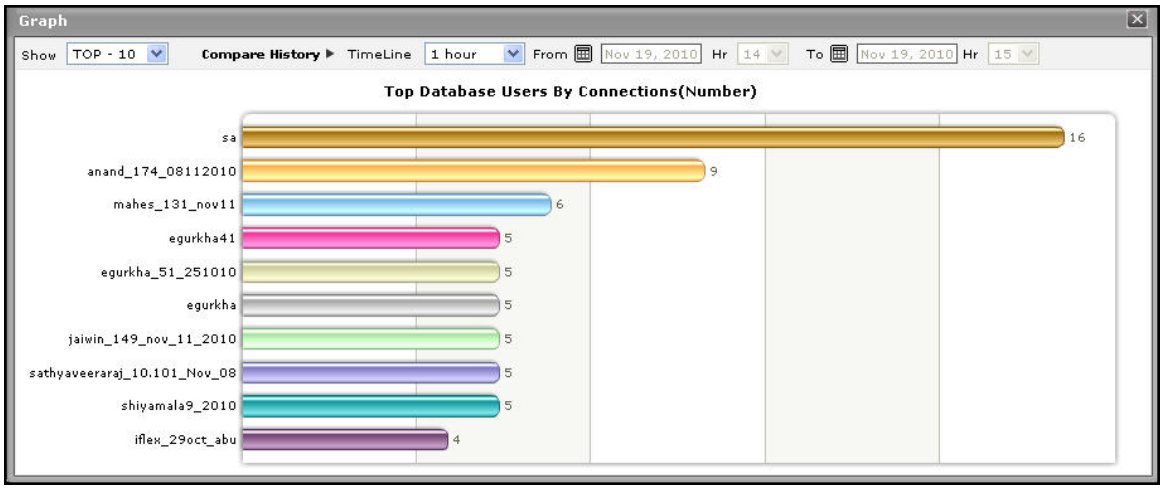


Figure 3.56: The expanded Comparison graph in the SQLProcesses dashboard

3. Though the enlarged graph lists the database users, by default, you can customize the enlarged graph to display the details of only a few of the best/worst-performing processes by picking a **TOP-N** or **LAST-N** option from the **Show** list in Figure 3.56.
4. Another default aspect of the enlarged graph is that it pertains to the current period only. Sometimes however, you might want to know what occurred during a point of time in the past; for instance, while trying to understand the reason behind a sudden slowdown in a particular process on a particular day last week, you might want to first determine which process has behaved abnormally on the same day. To figure this out, the enlarged graph allows you to compare the historical performance of the processes. For this purpose, click on the **Compare History** link in Figure 3.56 and select the **TimeLine** of your choice.
5. The **History** tab page below, by default, provides a series of measure graphs that reveal how the process has been performing over the default duration of the last 24 hours. If there is a sudden slowdown in the process, it could indicate that the server is experiencing issues with the concerned process. In such a case, a look at these measure graphs will help you figure out when exactly the bottleneck surfaced - did it happen suddenly or is it a condition that has become worse with time?
6. The default duration of 24 hours can be overridden using the procedure discussed below:
   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.

- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

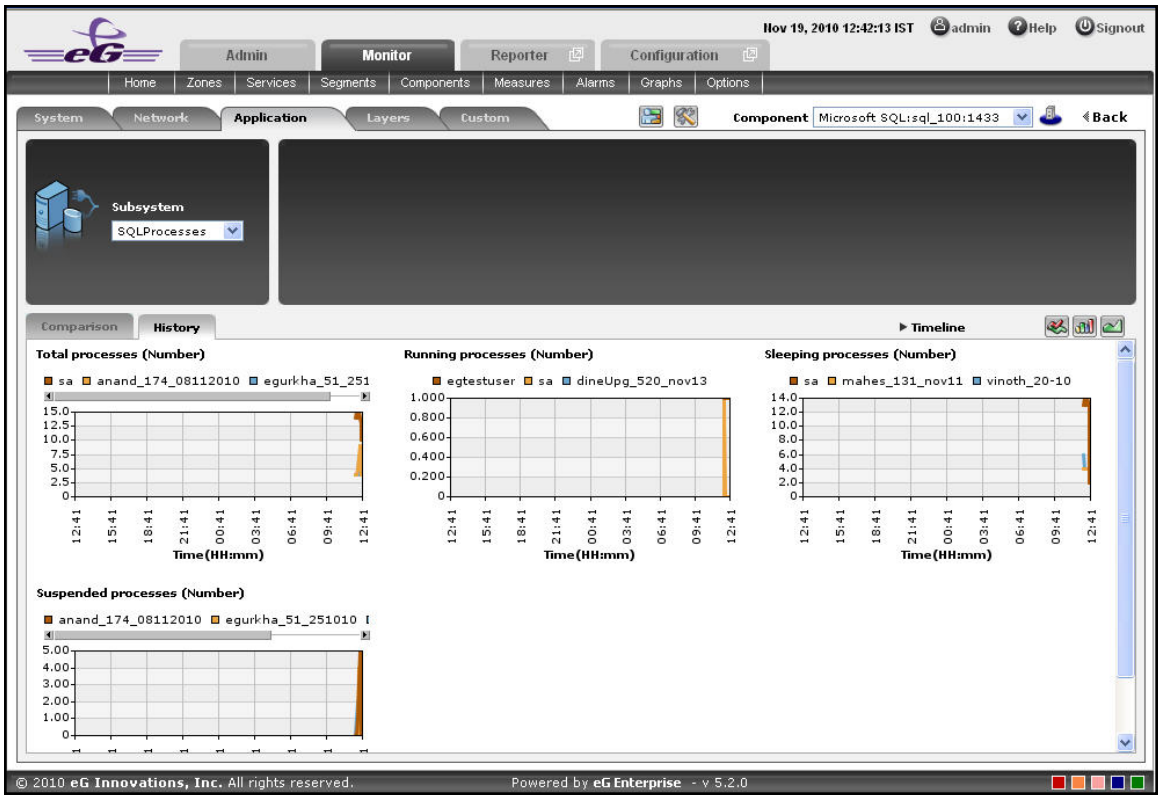


Figure 3.57: Figure 3.56: The History tab page of the SQLProcesses dashboard

7. If need be, you can even alter the timeline of all these measure graphs so that you can analyze performance across days and weeks; for this, simply click the **Timeline** link at the right, top corner of the **History** tab page and change the timeline for the graphs using the calendar that pops out. To change the timeline of a single graph alone, simply click on that graph to enlarge it, and then modify the **Timeline** of the graph in the enlarged mode. In the enlarged mode, you can even change the dimension of the measure graph (**3D** / **2D**).
8. To determine the service level achievements / slippages of the process, you need to view summary graphs of the measures and not the default measure graphs. For this, just click on the icon at the right, top corner of the **History** tab page.
9. The summary graphs reveal the percentage of time the process experienced problems in the database. Besides revealing the efficiency of your administrative staff in recognizing bottlenecks and mitigating them, these summary graphs also indicate whether the class loader has been able to maintain the assured

performance levels during the default duration of 24 hours.

To override this default duration, follow the steps below:

- Click on the  icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

10. In case of the summary graphs too, you can change the **Timeline** of all graphs by clicking on the **Timeline** link at the right, top corner of the **History** tab page. To alter the timeline of a single graph, here again, you will have to click on that graph, enlarge it, and modify the timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode.
11. To analyze past trends in the behavior of the processes, click on the  icon at the right, top corner of the **History** tab page. These trend graphs, by default, plot the minimum and maximum values that every measure registered during each hour of the last 24 hours (by default). By carefully observing these past trends, you can effectively analyze the workload of the process, predict future workloads accordingly, and suggest measures to enhance the efficiency of the process. Here again, you can change the timeline of all graphs using the **Timeline** link, or just a particular graph by clicking on it and enlarging it.
12. For changing the default duration (of 24 hours) of the trend graphs, do the following:
    - Click on the  icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.
13. In addition, when a trend graph is enlarged, it is not just the **Timeline** that you can modify. The **Duration** of the graph can also be altered. By default, trend graphs reveal only the hourly trends in performance. By picking the relevant option from the **Duration** list, you can ensure that the trend graph in question plots daily/monthly trend values instead. Also, in the enlarged mode, the **Graph type** can also be modified. Since the default **Graph type** is **Min/Max**, the trend graph, by default, reveals the minimum and maximum values registered by a measure. If need be, you can select the **Avg** or **Sum** option from the **Graph type** list to plot average trend values of a measure or sum of trends (as the case may be) in the graph.

    Note:

    In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.
14. At any point in time, you can switch to the measure graphs by clicking on the  button.
15. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of

measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:

- Click the  button at the top of the dashboard.
- The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **SQLProcesses** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
- The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
- To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
- Next, select the **Measure** of interest.
- Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
- This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.5 SQLDatabases

Select the **SQLDatabases** option from the **Subsystem** list to know how efficiently the databases are used by the MS SQL application and how well the database has been responding to the queries from other applications. Upon selection, Figure 3.58 will appear.

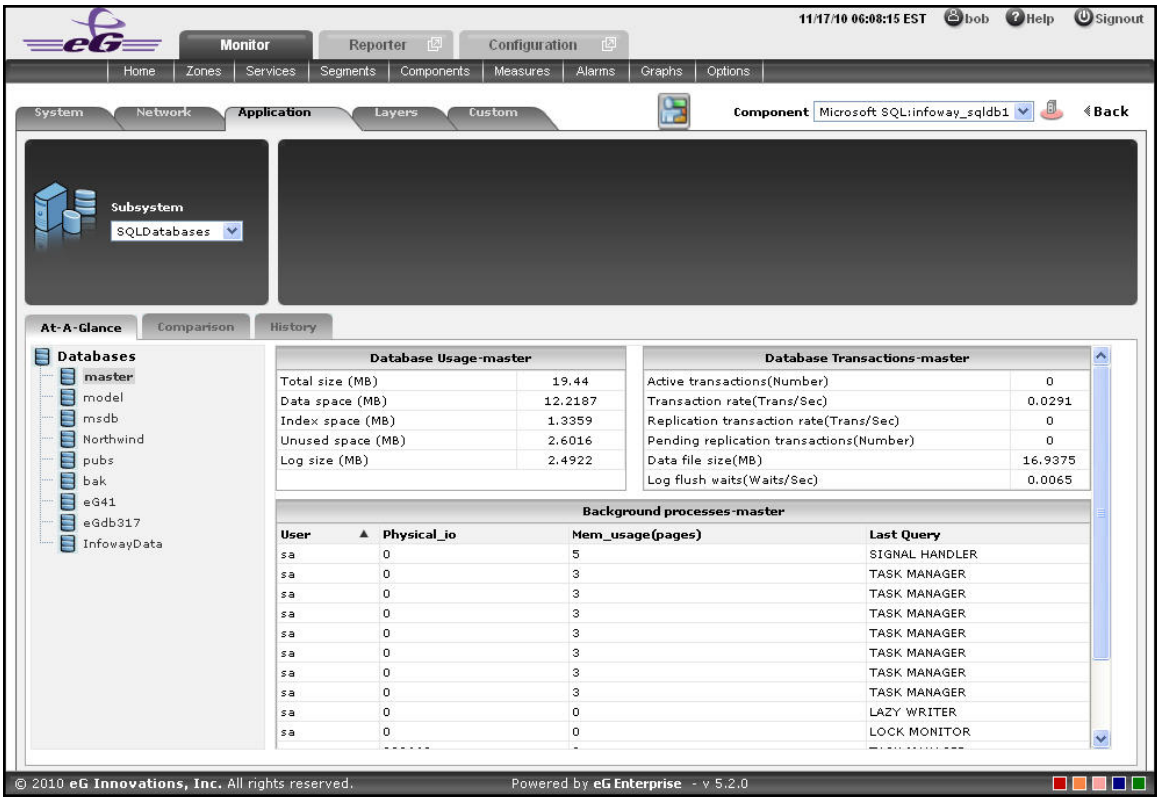


Figure 3.58: The At-A-Glance tab page of the SQLDatabases Dashboard

The contents of this dashboard are as follows:

1. The **At-A-Glance** tab page lists the databases that are available at present in the MS SQL application, in the **Databases** section. Upon selecting a database, the detailed measures corresponding to that particular database is available in a context-sensitive right panel. For instance, if master database is selected, then in the right panel, the **Database Usage** section will provide the usage details like **Total size**, etc. Also the **Database Transactions** section provides the relevant transaction measures The **Background processes** section will list out all the background process available for the users who are accessing that particular database. By default, the background process list provided by this section is sorted in the alphabetical order of the User. If need be, you can change the sort order so that the processes are arranged in, say, the descending order of values displayed in the **Physical_io** column - this column displays the physical io location of each user connected to the database. To achieve this, simply click on the column heading – **Physical_io**. Doing so tags the **Physical_io** label with a **down arrow** icon - this icon indicates that the background process list is currently sorted in the descending order of physical io location. To change the sort order to 'ascending', all you need to do is just click again on the **Physical_io** label or the **down arrow** icon. Similarly, you can sort the process list based on any column available in the **Background processes** section. Likewise the right panel may consist of **Running processes** and **Sleeping processes** sections, if those particular processes are available for execution in the selected database. Similarly **CPU cycles rate** section may also be available for the databases. This section reveals the number of CPU cycles taken by the server for each host available in the target MS SQL application. The columns available in this section can also be sorted in the same manner as that of the **Background processes** section.

2. The Comparison tab page that follows the At-A-Glance tab page provides a series of top-10 charts, using which you can isolate the databases that are leading the lot in the following fields: Size, Log Size and Active transactions. This default list of fields (i.e., measures) for top-n chart generation can be overridden by following the steps discussed below:

   - Click on the  icon at the top of the **Application Dashboard**. In the **Dashboard Settings** window that appears, select **Application** from the **Module** list, and **SQLDatabases** from the **Sub-System** list.
   - To add new measures for which top-n graphs are to be displayed in the **Comparison** tab page, first, pick the **Comparison Graph** option from the **Add/Delete Measures for** list. Upon selection of this option, the pre-configured measures for comparison graphs will appear in the **Existing Value(s)** list.
   - Next, select the **Test** that reports the said measure, pick the measure of interest from the **Measures** list, provide a **Display** name for the measure, and click the **Add** button to add the chosen measure to the **Existing Value(s)** list.
   - If you want to delete one/more measures for which comparison graphs pre-exist in the **Comparison** tab page, then, as soon as you choose the **Comparison Graph** option from the **Add/Delete Measures for** list, pick any of the displayed measures from the **Existing Value(s)** list, and click the **Delete** button.
   - Finally, click the **Update** button to register the changes.

   Note:

   Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

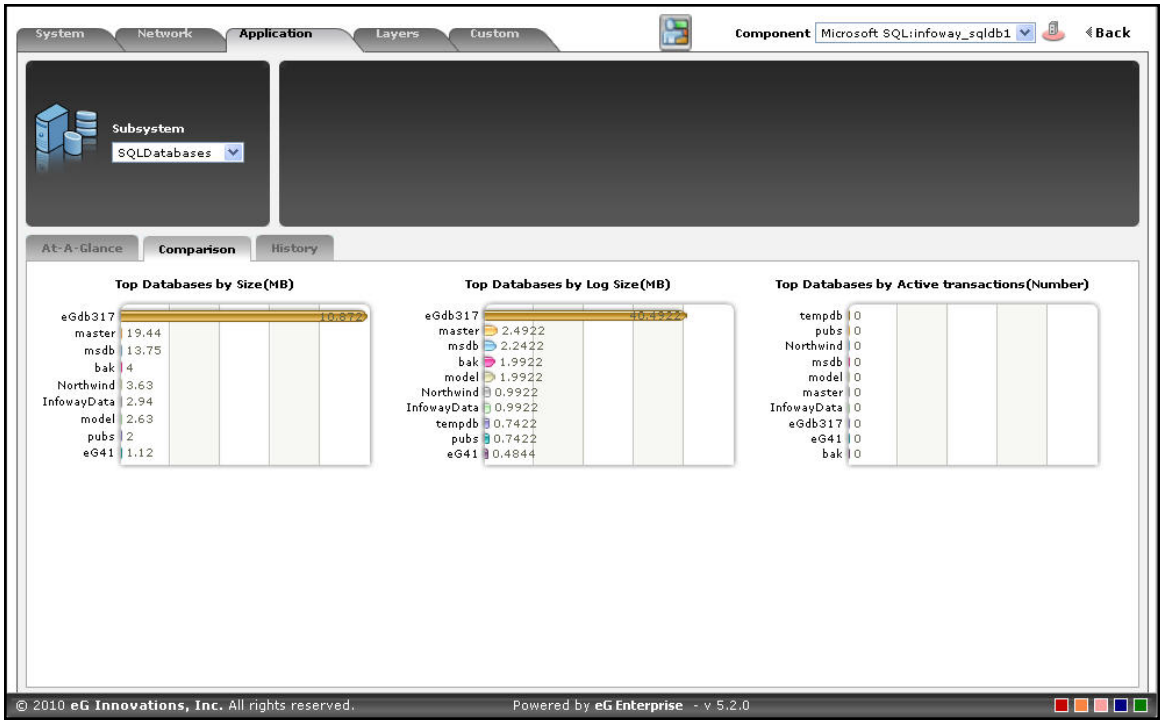


Figure 3.59: The Comparison tab page of the SQLDatabases dashboard

3. To view the complete list of databases, simply click on the corresponding graph in Figure 3.59. This enlarges the graph as depicted by Figure 3.60.

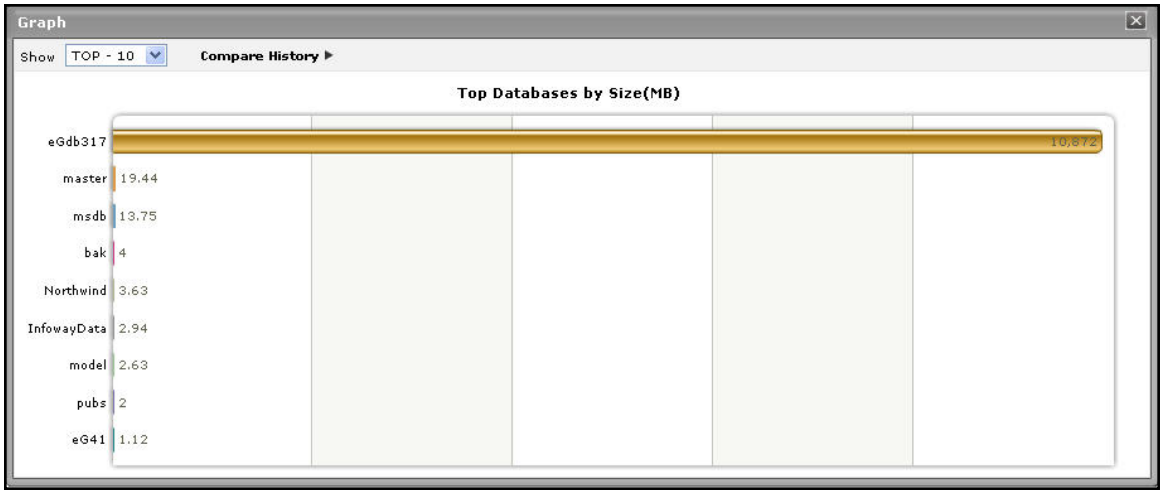


Figure 3.60: The expanded top-n graph in the Comparison tab page of the SQLDatabases Dashboard

4. Though the enlarged graph lists all the databases by default, you can customize the enlarged graph to display the details of only a few of the best/worst-performing databases by picking a **TOP-N** or **LAST-N** option from the **Show** list in Figure 3.60.
5. Another default aspect of the enlarged graph is that it pertains to the current period only. Sometimes however, you might want to know what occurred during a point of time in the past; for instance, while trying to understand the reason behind a sudden increase in the Size of the databases on a particular day last week, you might want to first determine which database has behaved abnormally on the same day. To figure this out, the enlarged graph allows you to compare the historical performance of databases. For this purpose, click on the **Compare History** link in Figure 3.60 and select the **TimeLine** of your choice.
6. The **History** tab page below, by default, provides a series of measure graphs that reveal how well the databases have been performing over the default duration of the last 24 hours. If the performance of the databases dramatically decreases, it could indicate that the databases are experiencing performance issues. In such a case, a look at these measure graphs will help you figure out when exactly the bottleneck surfaced - did it happen suddenly or is it a condition that has become worse with time?
7. The default duration of 24 hours can be overridden using the procedure discussed below:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

   Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the

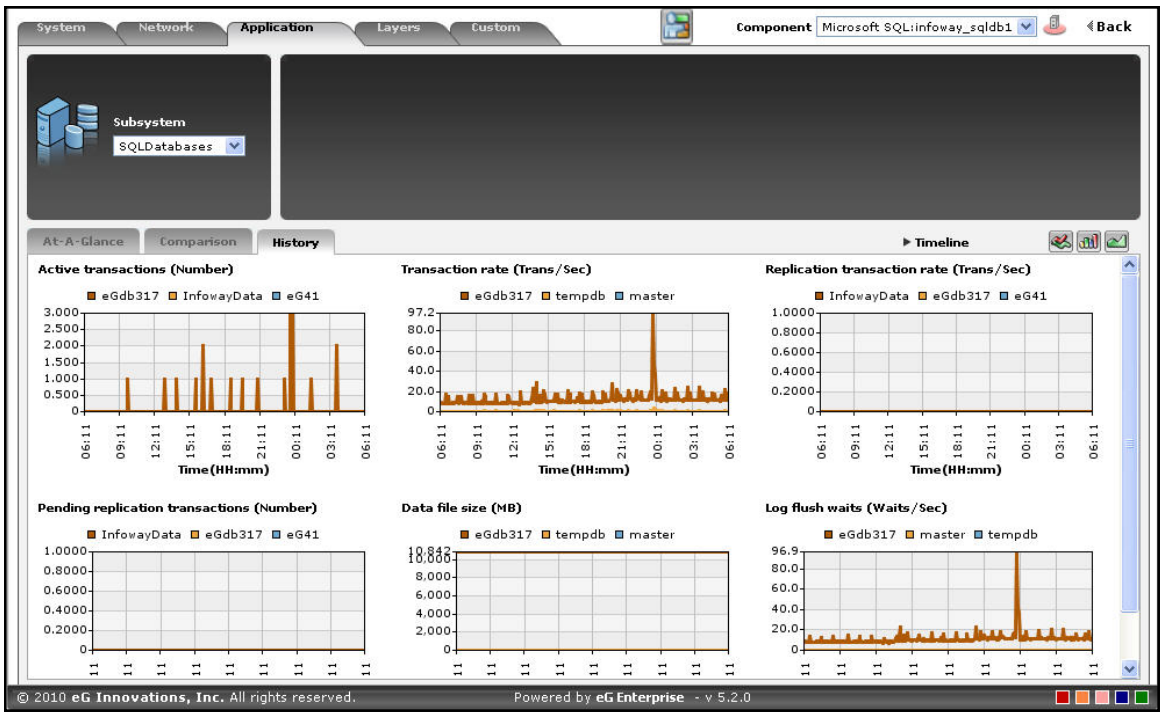


Figure 3.61: The History tab page of the SQLDatabases dashboard

8. If need be, you can even alter the timeline of all these measure graphs so that you can analyze performance across days and weeks; for this, simply click the **Timeline** link at the right, top corner of the **History** tab page and change the timeline for the graphs using the calendar that pops out. To change the timeline of a single graph alone, simply click on that graph to enlarge it, and then modify the **Timeline** of the graph in the enlarged mode. Though the enlarged graph lists all the databases by default, you can customize the enlarged graph to display the details of only a few of the best/worst-performing databases by picking a **TOP-N** or **LAST-N** option from the **Show** list. In the enlarged mode, you can even change the dimension of the measure graph (**3D / 2D**).

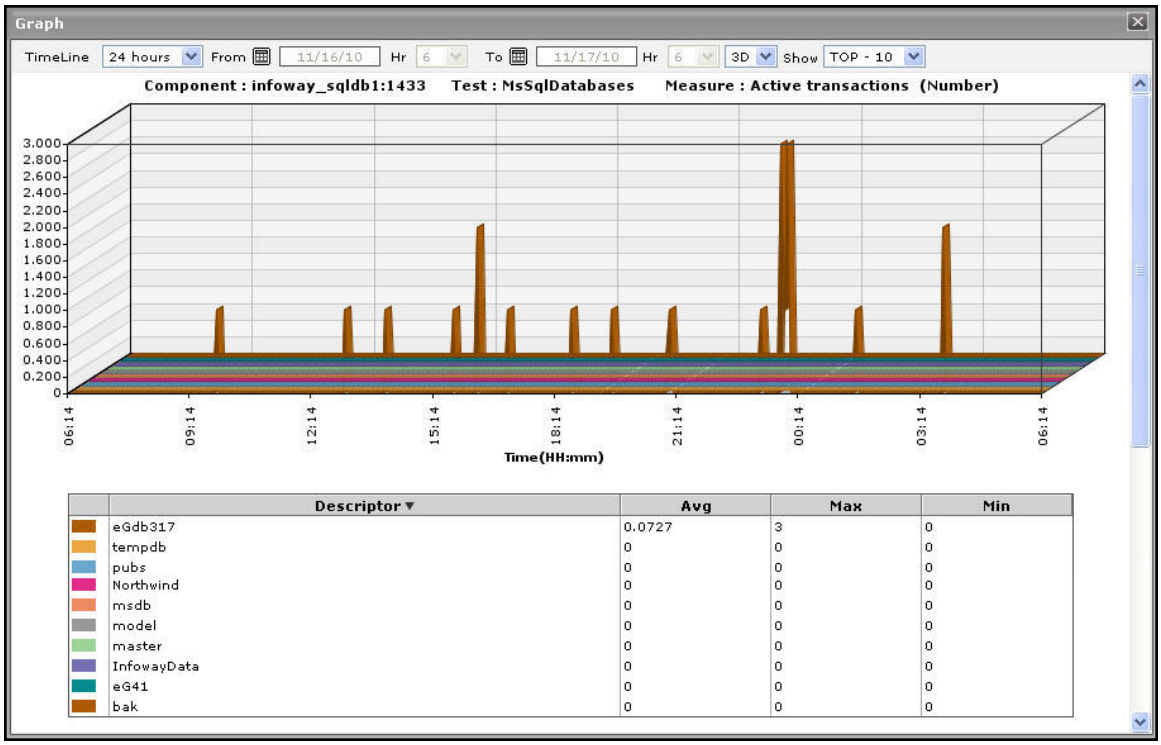


Figure 3.62: An enlarged measure graph in the History tab page of the SQLDatabases dashboard

9. To determine the service level achievements of the databases, you need to view summary graphs of the measures and not the default measure graphs. For this, just click on the icon at the right, top corner of the **History** tab page. Figure 3.63 then appears.

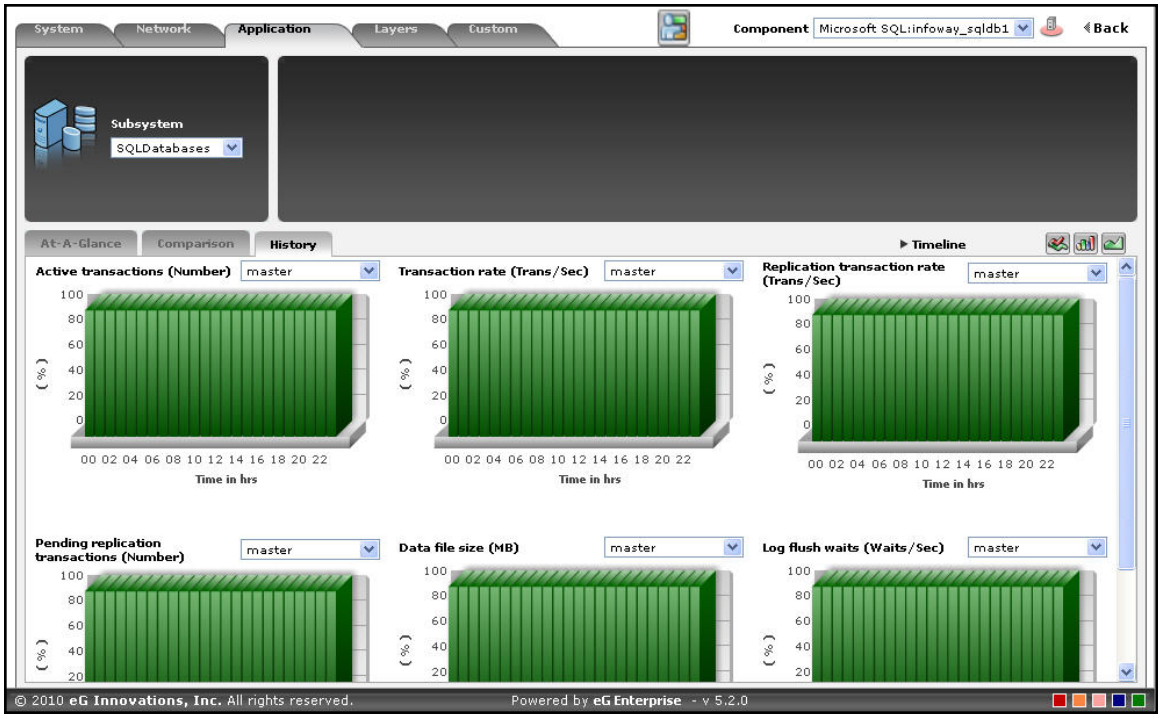


Figure 3.63: Summary graphs displayed in the SQLDatabases Dashboard

10. The summary graphs displayed in Figure 3.63 reveal the percentage of time the MS SQL application experienced problems in one of its databases. Besides revealing the efficiency of your administrative staff in recognizing bottlenecks and mitigating them, these summary graphs also indicate whether the databases has been able to maintain the assured performance levels during the default duration of 24 hours.
11. To override this default duration, follow the steps below:
    - Click on the  icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.
12. In case of the summary graphs too, you can change the **Timeline** of all graphs by clicking on the **Timeline** link at the right, top corner of the **History** tab page. To alter the timeline of a single graph, here again, you will have to click on that graph, enlarge it, and modify the timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode.
13. To analyze past trends in the performance of the databases, click on the  icon at the right, top corner of the **History** tab page. Figure 3.64 will then appear.

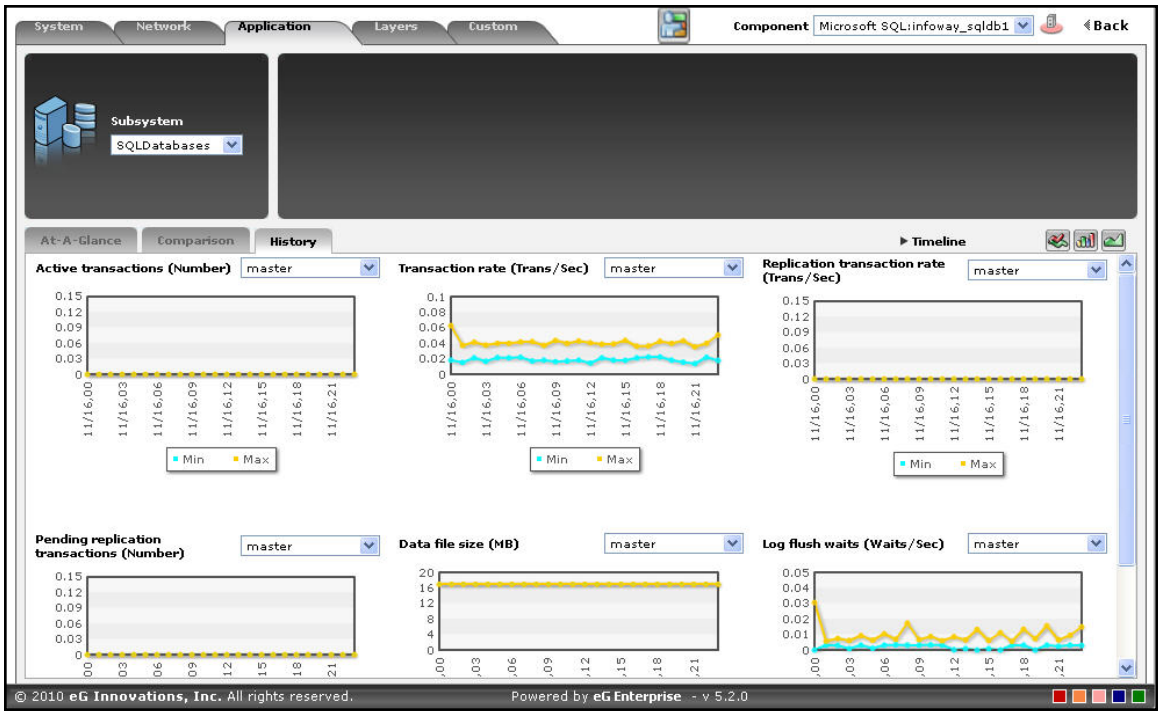


Figure 3.64: Trend graphs displayed in the SQLDatabases Dashboard

14. These trend graphs, by default, plot the minimum and maximum values that every measure registered during each hour of the last 24 hours (by default). Using such graphs, you can accurately point to the time windows during which there was a lull in the transaction of the selected database. Here again, you can change the timeline of all graphs using the **Timeline** link in Figure 3.64, or just a particular graph by clicking on it and enlarging it.
15. For changing the default duration (of 24 hours) of the trend graphs, do the following:
    - Click on the  icon at the top of the **Application Dashboard**.
    - In the **Dashboard Settings** window that appears, select **Trend Graph** from the **Default Timeline for** list.
    - Then, choose a **Timeline** for the graph.
    - Finally, click the **Update** button.
16. In addition, when a trend graph is enlarged, it is not just the **Timeline** that you can modify. The **Duration** of the graph can also be altered. By default, trend graphs reveal only the hourly trends in performance. By picking the relevant option from the **Duration** list, you can ensure that the trend graph in question plots daily/monthly trend values instead. Also, in the enlarged mode, the **Graph type** can also be modified. Since the default **Graph type** is **Min/Max**, the trend graph, by default, reveals the minimum and maximum values registered by a measure. If need be, you can select the **Avg** or **Sum** option from the **Graph type** list to plot average trend values of a measure or sum of trends (as the case may be) in the graph.

    Note:

    In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.
17. At any point in time, you can switch to the measure graphs by clicking on the  button.
18. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
    - Click the  button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **SQLDatabases** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
    - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
    - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
    - Next, select the **Measure** of interest.
    - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.

- This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

Note:

Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.6 SQLApplications

Select the **SQLApplications** option from the **Subsystem** list to know how well the applications are being used by the MS SQL application, Upon selecting this **Subsystem**, Figure 3.65 will appear.

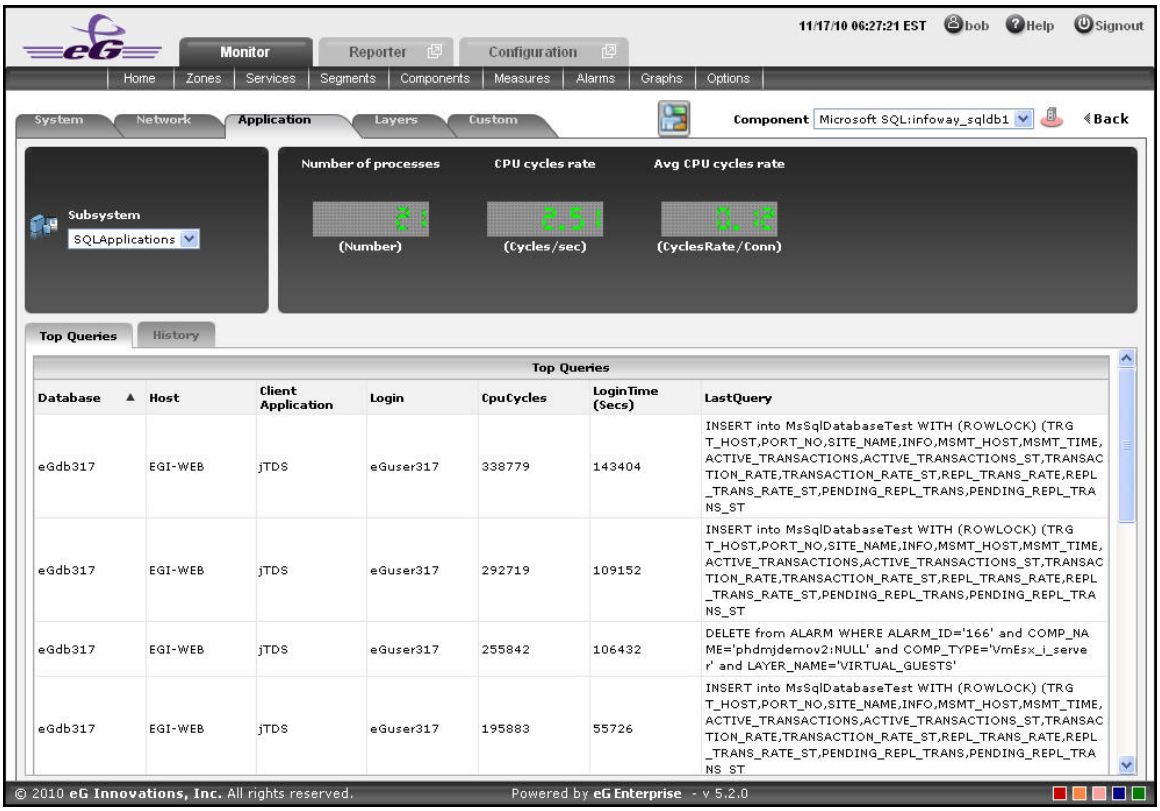


Figure 3.65: The SQLApplications Dashboard

The contents of the **SQLApplications** dashboard are as follows:

1. For an easy and single glance view of certain measures like Number of processes, CPU cycle rate and Avg CPU cycles rate, a digital graph section is included. When a digital graph is clicked, the corresponding layer-test combination which reports that particular measure will be displayed from the layer model page of the MS SQL application.
2. From the **Top Queries** tab page, you can infer the queries that have been made to the databases from the client application. By default, the queries listed in this tab page are sorted in alphabetical order of the

Database. If need be, you can change the sort order so that the databases are arranged in, say, the descending order of values displayed in the **CPUCycles** column. To achieve this, simply click on the column heading – **CPUCycles**. Doing so tags the **CPUCycles** label with a **down arrow** icon - this icon indicates that this tab page is currently sorted in the descending order of the CPU cycles. To change the sort order to 'ascending', all you need to do is just click again on the **CPUCycles** label or the **down arrow** icon. Similarly, you can sort the process list based on any column available in this tab page.

3. The **History** tab page, by default, displays time-of-day graphs revealing how well the application has been performing over a default period of 24 hours. If the eG agent reports any abnormal behavior of the MS SQL application, these graphs will help determine when exactly in the last 24 hours the abnormality occurred. This default duration of 24 hours can be overridden using the following steps:

   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Figure 3.66: The history tab page of the MS SQL Applications Dashboard

4. A careful study of this graph over time periods longer than 24 hours, can reveal intermittent breaks (if any) in the number of processes and the CPU cycle rate of the databases. To ensure that all graphs plot values for longer time periods, click on the **Timeline** link at the right, top corner of the **History** tab page, and then change the timeline using the calendar that pops out. To modify the timeline for a particular graph alone, click on the graph to enlarge it, and alter the timeline in the enlarged mode. Besides the timeline, you can even change the graph dimension (**3D / 2D**) in the enlarged mode.

5. Sometimes, you might have to periodically determine the percentage of time for which the MS SQL application experienced problems relating to the databases. To determine such problems, summary graphs of the SQL applications measures are useful. To view summary graphs in the **History** tab page, click on the  icon at the right, top corner of the **History** tab page. These summary graphs reveal the

percentage of time during the last 24 hours (by default) the MS SQL application has experienced issues related to the SQLapplications. To override this default timeline, do the following:

- Click on the  icon at the top of the **Application Dashboard**.
- In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
- Then, choose a **Timeline** for the graph.
- Finally, click the **Update** button.

6. To perform the summary analysis over a broader time window, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode. Here again, the graph dimension (**3D / 2D**) can be altered.
7. Similarly, you can analyze uptime trends by viewing trend graphs in the **History** tab page. For this, click on the  icon at the right, top corner of the tab page. These trend graphs, by default, plot the minimum and maximum values registered by every SQL application-related measure during every hour for the last 24 hours. The default duration of 24 hours can be overridden using the procedure discussed below:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

   To perform trend analysis over a longer time span, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. In addition to the timeline, the graph dimension (**3D / 2D**), the graph **Duration**, and the **Graph type** can also be changed in the enlarged mode. By default, the graph **Duration** is **Hourly**, indicating that trend graphs plot hourly trend values by default. To ensure that these graphs plot the daily/monthly trend values instead, select the relevant option from the **Duration** list. Similarly, as already mentioned, trend graphs plot only the minimum and maximum values registered by a measure during the specified timeline. Accordingly, the **Graph type** is set to **Min/Max** by default in the enlarged mode. If you want the trend graph to plot the average trend values instead, set the **Graph type** to **Avg**. On the other hand, to configure the trend graph to plot the sum of trends set the **Graph type** to **Sum**.

   Note:

   In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

8. At any point in time, you can switch to the measure graphs by clicking on the  button.

9. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:

    - Click the  button at the top of the dashboard.
    - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **SQLApplication** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
    - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
    - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
    - Next, select the **Measure** of interest.
    - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
    - This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

    Note:

    Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.7.7 SQLService

The **SQLService** option is picked from the **Subsystem** list to know about the overall performance of the MS SQL application such as server health, session activity, access capacity of the server etc, in detail. Upon selecting this **Subsystem**, Figure 3.67 will appear.

Figure 3.67: The SQLService Dashboard

The contents of the **SQLService** dashboard are as follows:

1. A digital display section for some critical session activity and access capability measures is included for an easy and single glance view. When a digital display is clicked, the corresponding layer-test combination which reports that particular measure will be displayed from the layer model page of the MS SQL application.
2. The **History** tab page as shown in Figure 3.67, by default, displays time-of-day graphs revealing how well the application has been performing over a default period of 24 hours. If the eG agent reports any abnormal behavior of the MS SQL application, these graphs will help determine when exactly in the last 24 hours the abnormality occurred. This default duration of 24 hours can be overridden using the following steps:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **History Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

   Note:

   Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard**

Settings window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the button will not appear.

3. A careful study of this graph over time periods longer than 24 hours, can reveal intermittent breaks (if any) in the session activity and access capability measures of this MS SQL application. To ensure that all graphs plot values for longer time periods, click on the **Timeline** link at the right, top corner of the **History** tab page, and then change the timeline using the calendar that pops out. To modify the timeline for a particular graph alone, click on the graph to enlarge it (see Figure 3.68), and alter the timeline in the enlarged mode. Besides the timeline, you can even change the graph dimension (**3D / 2D**) in the enlarged mode.

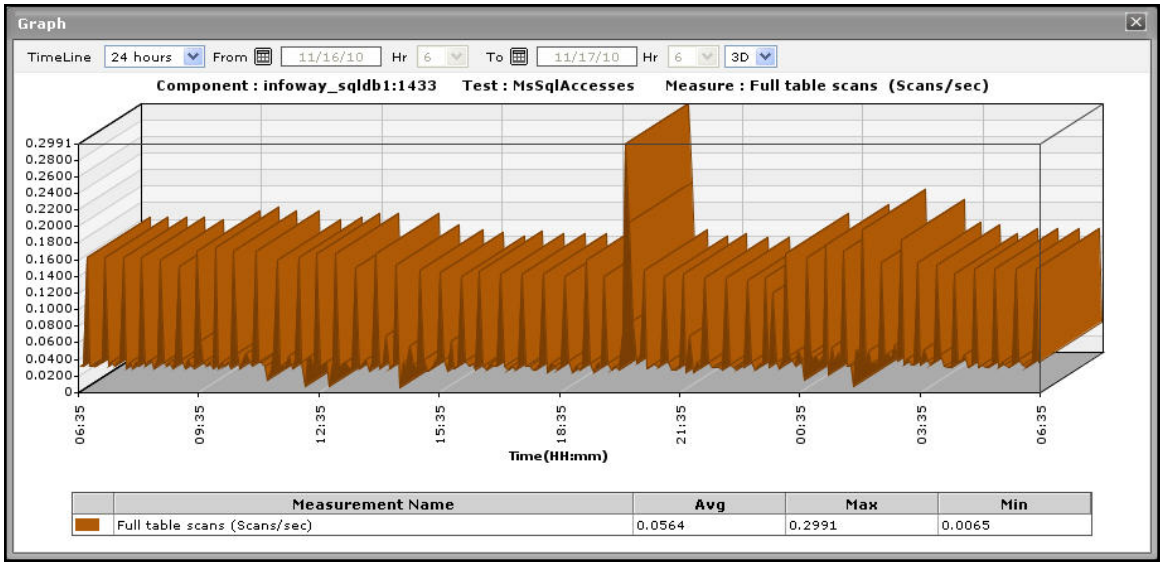


Figure 3.68: The enlarged history graph of the SQLService dashboard

4. Sometimes, you might have to periodically determine the percentage of time for which the MS SQL application experienced problems relating to the databases. To determine such problems, summary graphs are useful. To view summary graphs in the **History** tab page, click on the icon at the right, top corner of the **History** tab page. These summary graphs reveal the percentage of time during the last 24 hours (by default) the MS SQL application has experienced issues related to the SQL service. To override this default timeline, do the following:

   - Click on the icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.

Figure 3.69: The summay graphs for the SQLService dashboard

5. To perform the summary analysis over a broader time window, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. Also, by default, hourly summaries are plotted in the summary graph; you can configure these graphs to plot daily/monthly summaries instead by picking the relevant option from the **Duration** list in the enlarged mode. Here again, the graph dimension (**3D / 2D**) can be altered.
6. Similarly, you can analyze uptime trends by viewing trend graphs in the **History** tab page. For this, click on the  icon at the right, top corner of the tab page. These trend graphs, by default, plot the minimum and maximum values registered by every SQL application-related measure during every hour for the last 24 hours. The default duration of 24 hours can be overridden using the procedure discussed below:
   - Click on the  icon at the top of the **Application Dashboard**.
   - In the **Dashboard Settings** window that appears, select **Summary Graph** from the **Default Timeline for** list.
   - Then, choose a **Timeline** for the graph.
   - Finally, click the **Update** button.
7. To perform trend analysis over a longer time span, click on the **Timeline** link at the right, top corner of the **History** tab page and change the timeline; this will alter the timeline for all the graphs. To change the timeline of a particular graph alone, click on the graph to enlarge it, and then alter its timeline. In addition to the timeline, the graph dimension (**3D / 2D**), the graph **Duration**, and the **Graph type** can also be changed in the enlarged mode. By default, the graph **Duration** is **Hourly**, indicating that trend graphs plot hourly

trend values by default. To ensure that these graphs plot the daily/monthly trend values instead, select the relevant option from the **Duration** list. Similarly, as already mentioned, trend graphs plot only the minimum and maximum values registered by a measure during the specified timeline. Accordingly, the **Graph type** is set to **Min/Max** by default in the enlarged mode. If you want the trend graph to plot the average trend values instead, set the **Graph type** to **Avg**. On the other hand, to configure the trend graph to plot the sum of trends set the **Graph type** to **Sum**.

Note:

In case of descriptor-based tests, the **Summary** and **Trend** graphs displayed in the **History** tab page typically plot the values for a single descriptor alone. To view the graph for another descriptor, pick a descriptor from the drop-down list made available above the corresponding summary/trend graph.

8. At any point in time, you can switch to the measure graphs by clicking on the  button.
9. Typically, the **History** tab page displays measure, summary, and trend graphs for a default set of measures. If you want to add graphs for more measures to this tab page or remove one/more measures for which graphs pre-exist in this tab page, then, do the following:
   - Click the  button at the top of the dashboard.
   - The **Dashboard Settings** window then appears. From the **Module** list of Figure 3.45, pick **Application**, choose **SQLService** as the **Sub-System**, and then, select **History Graph** from the **Add/Delete Measures for** list.
   - The measures for which graphs pre-exist in the **History** tab page will be automatically displayed in the **Existing Value(s)** list. To delete a measure, and in effect, its corresponding graph as well, select the measure from the **Existing Value(s)** list, click the **Delete** button, and then click the **Update** button.
   - To add a new graph, first, pick the **Test** that reports the measure for which a graph is to be generated.
   - Next, select the **Measure** of interest.
   - Provide a **Display** name for the measure. Then, click the **Add** button to add the measure to the **Existing Values(s)** list. Finally, click the **Update** button.
   - This will add a new measure, summary, and trend graph for the chosen measure to the **History** tab page.

   Note:

   Only users with **Admin** or **Supermonitor** privileges can enable/disable the system, network, and application dashboards, or can customize the contents of such dashboards using the **Dashboard Settings** window. Therefore, whenever a user without **Admin** or **Supermonitor** privileges logs into the monitoring console, the  button will not appear.

3.8 Troubleshooting

If the measurements of a Microsoft SQL Server are not showing up, check the following:

- Is the internal agent running on the Microsoft SQL server?
- Is the user / password provided for the Microsoft SqlNet test valid? Does the user specified have the privileges mentioned in Page 180?
- eG agents use Windows perfmon counters to monitor Microsoft SQL servers. If these perfmon counters are not enabled, then the eG agents monitoring the Microsoft SQL server will not be able to generate measurements. In order to enable the perfmon counters of the Microsoft SQL server, do the following:
  - Stop the **MSSQLSERVER** service, if running. To do this, first, open the **Services** window using the menu sequence: Start -> Programs -> Administrative Tools -> Services. From the right panel of the window, select the **MSSQLSERVER** service, right-click on it and select **Stop** from the shortcut menu that pops out.
  - Next, at the command prompt, move to the **<MSSQL_HOME_DIR>\binn** directory and then, type the command **lodctr sqlctr.ini**. This file contains information on all the counters that correspond to the **MSSQLSERVER** service. Loading this file activates the counters within.
  - Finally, restart the machine.
  - Upon restarting, all counters corresponding to the **MSSQLSERVER** service will be enabled.
- If the Network test alone is working, then it could be because the internal agent is not running.
- If the Network test is not gathering measurements, then check whether the external agent is running.
- If the SQL Network test is not reporting measurements, verify whether the username and password were configured via the admin interface.
- If the SQL Network test shows that availability is 0, but the SQL server is up and running, then check whether the right user name and password were configured.

4

Conclusion

This document has described in detail the monitoring paradigm used and the measurement capabilities of the eG Enterprise suite of products with respect to **Microsoft SQL** servers. For details of how to administer and use the eG Enterprise suite of products, refer to the user manuals.

We will be adding new measurement capabilities into the future versions of the eG Enterprise suite. If you can identify new capabilities that you would like us to incorporate in the eG Enterprise suite of products, please contact support@eginnovations.com. We look forward to your support and cooperation. Any feedback regarding this manual or any other aspects of the eG Enterprise suite can be forwarded to feedback@eginnovations.com.