SQL Server Findings

This section includes the following topics:

- About the Findings view
- About the Highlights area
- About the What to do next area
- About the Advice area
- How to investigate findings
- About Statement findings
- About Object findings
- About Instance findings
- About Storage findings

About the Findings view

The Findings view displays recommendations that can be used to create a better execution plan and improve the performance of a statement. The Findings vary according to the type of operation in a statement, the Precise product and the technology.

About the Highlights area

The Highlights area displays a brief description of the findings for the selected type of operation.

About the What to do next area

The What to do next area displays one or more recommended steps to identify the cause of the problem. Carefully review all data for the finding before continuing.

About the Advice area

The Advice area displays one or more recommended options to resolve or reduce the problem for the selected finding. Carefully review all data for the finding and then perform the advice that best suits your needs.

How to investigate findings

When you start investigating the findings, it is good practice to start with the finding that has the highest severity ranking in the Findings table.

To investigate a finding

1. Identify the finding with the highest severity ranking in the Findings table.
2. Select the finding type to view additional information on the selected type of operation.
3. Read the Highlights and What to do next areas for the finding.
4. After you have studied all of the information provided, read the Advice area and perform the recommendation that best suits your needs.
5. Follow up on performance to verify that the problem was resolved.

About Statement findings

The following statement findings can help you tune your system:

- Heavy Statement
- Heavy Collapsed Statement
- Major Statement in Batch
- Heavy Operators
- Missing Indexes
- Missing Statistics
- Table Schema Change May Increase Its Accessing Time
- Statement Is Not Scalable
- Table Growth May Increase Its Accessing Time
- Increase in Resource Consumption
- Statement or Batch Was Locked
- Statement Activity Consistently High

Heavy Statement

The statement is a major consumer of MS-SQL resources. By tuning the statement, you can free resources needed by other statements and processes.

Table 1 Heavy Statement findings

Column	Description

What to do next	Try to determine what is causing the statement's high resource consumption. In the SQL tab, examine the text of the relevant statement, and its findings, execution plan, change data and statistics.
Advice	The following scenarios indicate what factors can lead to heavy resource consumption and what steps you can take: • A heavy operator was identified. Use the SQL findings to identify the heaviest operator. • A missing index was identified (SQL Server 2005 only). The SQL Server optimizer identified missing indexes when the access plan was created. Detailed index information can be found in the Recommend tab in the SQL tab. • Missing statistics were identified. The SQL Server optimizer identified missing statistics when the access plan was created. Detailed information on missing statistics for this statement can be found in the Recommend tab in the SQL tab. • An increase of In MS-SQL time was identified. Use Precise for SQL Server to try and locate the cause. Check schema changes, major table growth and scalability changes.

Heavy Collapsed Statement

A collapsed statement includes several statements that use the same text, but not the same constants.

A set of collapsed statements constitute a major consumer of MS-SQL resources. By tuning the set of statements, you can free resources needed by other statements and processes.

Table 2 Heavy collapsed statement findings

Column	Description
What to do next	In the SmarTune tab, examine the text of the relevant statements. Check their scalability and associated list of statements. Select the statement you want to analyze, and launch to the SQL tab with the statement in context. Examine its execution plan, change data and statistics to determine what is causing its high resource consumption.

Advice	The following scenarios indicate what factors can lead to heavy resource consumption and what steps you can take:
	• A scalability issue was raised. Check performance over time in the Over Time tab.
	• In the Statements tab, tune a statement by doing one of the following:
	  ◦ If a heavy operator was identified, use the SQL findings to identify the heaviest operator.
	  ◦ If a missing index was identified (SQL Server 2005 only), this means that the SQL Server optimizer identified a missing index when the access plan was created. Detailed index information can be found in the Recommend tab in the SQL tab.
	  ◦ If missing statistics were identified this means that the SQL Server optimizer identified missing statistics when the access plan was created. Detailed information on missing statistics for the collapsed statement can be found in the Recommend tab in the SQL tab.
	  ◦ If an increase in MS-SQL time was identified, use Precise for SQL Server to try and locate the cause. Check schema changes, major table growth and scalability changes.

Major Statement in Batch

The statement consumed more than 50% of MS-SQL batch resources. By tuning the statement, you can free resources needed by other statements and processes.

Table 3 Major Statement in Batch findings

Column	Description
What to do next	Tune the specific statement and examine its text, findings, execution plan, change data and statistics to determine what is causing the statement's high resource consumption.
Advice	The following scenarios indicate what factors can lead to heavy resource consumption and what steps you can take: • **A heavy operator was identified**. Use the SQL findings to identify the heaviest operator. • **A missing index was identified (SQL Server 2005 only)**. The SQL Server optimizer identified missing indexes when the access plan was created. Detailed index information can be found in the Recommend tab in the SQL tab. • **Missing statistics were identified**. The SQL Server optimizer identified missing statistics when the access plan was created. Detailed information on missing statistics for this statement can be found in the Recommend tab in the SQL tab. • **An increase of In MS-SQL time was identified**. Use Precise for SQL Server to try and locate the cause. Check schema changes, major table growth and scalability changes.

Heavy Operators

The statement or batch has an operator that shows a high cost percentage.

Table 4 Heavy Operators findings

Column	Description

What to do next	Perform one of the following options: • Click **Locate** to find the heavy operator and tune it. • Identify recommendations in the Recommend tab in the SQL tab. • In case of Hash or Sort operators examine internal wait for the statement.
Advice	One of the following operations (for example, table scan, index scan, clustered index scan or filter) was identified as a major cost indicator. Try to eliminate the major consuming operation by doing one of the following: For a table scan: • Eliminate a full table scan or cluster index scan. • Try to identify high selectivity columns for the statements or batches. Create an index matching the statement or batch's predicates. • Try to identify if there is a small set of columns that can be used in an index-only operation. • Partition the table according to the best predicates existing in the statements or batches. • Improve full table scan. Table fragmentation can affect overhead when a table scan or partial table scan is performed. Determine whether the table is heavily fragmented. If it is, defragment the table. For an index scan: • If a missing index finding is detected, launch to the Recommend tab. Otherwise try to identify where the matching level problem exists, indicating that the index column order does not match the statement or batch WHERE columns. Example: ◦ An index on A, B, C, while the statement or batch is "select … where A=3 and C=7". ◦ Many irrelevant index leaf pages will be read in this case. This can be avoided by either changing the index column sequence or adding columns to the index. ◦ Analyze the recommendations provided in the Recommend tab in the SQL tab. For a Filter: • Eliminate a filter by doing one of the following: ◦ Consider creating an index view. ◦ Consider performing an index on the computed column. For a Sort: • Try reducing the space or memory required for Sort operations by reducing the number of sorted columns or by filtering the rows to be sorted. • Try to find a way to eliminate the sort by using pre-sorted information like creating an index on the sorted columns. • Check the values of the *Minimum memory per query (KB)* and *Minimum memory for index create sorts (KB)* instance parameters. Use the *Memory for index create sorts (KB)* configuration to control the amount of memory used by index creation sorts. The *Minimum memory for index create sorts (KB)* configuration is self-configuring and should work in most cases without requiring adjustment. However, if you experience difficulties creating indexes, consider increasing the value of this option from its run value. Query sorts are controlled through the *Minimum memory per query (KB)*. For a Hash: • Hash join is used where the joined tables or row sets are large or where there aren't adequate indexes or the lack of indexes at all. • Verify that there are no missing indexes or the indexes are adequate. • Search for a missing WHERE clause or a missing condition in the WHERE clause. • Search for a non-sargeable expression. A non-sargeable expression is an expression preventing the optimizer of using the index in the ideal way or not at all like a function expression. • Check the values of the *Minimum Size of Server Memory (MB)* and *Max Size of Server Memory (MB)* instance parameters. For a Index Spool, Table Spool and Row Count Spool: • Spooling are internally temporary tables created by SQL Server on the tempdb database. Spooling may cause tempdb overhead and thus resulting in instance performance degradation caused by tempdb wait or tempdb major growth. • Try to eliminate spooling operations by rephrasing the query. • Try to reduce the number of logical reads or writes.

Missing Indexes

During SQL Server optimization, missing indexes were identified for the statement or batch. This means there can be one or more indexes, but they are not used because of a mismatch of column types. The SQL Server optimizer recommended creating indexes to improve the performance of the statement or batch. Detailed information regarding index recommendation for this statement or batch can be found in the Recommend tab of the SQL tab.

Table 5 Missing Indexes findings

Column	Description
What to do next	Examine the information displayed in the Recommend tab in the SQL tab to identify missing statistics and analyze index recommendations.

Advice	Perform one of the following options: • Examine the information displayed in the Recommend tab in the SQL tab to identify missing statistics and analyze index recommendations. • Use the What if feature in the Recommend tab to evaluate which statements or batches were affected, based on a large set of statements or batches that were active during the selected time frame. This feature lets you check whether it is possible to improve performance by creating one or several indexes.

Missing Statistics

The Missing Statistics warning was issued by the SQL Server optimizer when the access plan was created. This means that the SQL Server optimizer recommends creating and updating the statistics on objects accessed by a specific statement or batch. Detailed information on missing statistics for this statement or batch can be found in the Recommend tab in the SQL tab.

Table 6 Missing Statistics findings

Column	Description
What to do next	Examine the information displayed in the Recommend tab in the SQL tab to identify missing statistics and analyze index recommendations.
Advice	Perform the following options: • Examine the information displayed in the Recommend tab in the SQL tab to identify missing statistics and analyze index recommendations. • Use the What if feature in the Recommend tab to evaluate a missing index, based upon a large set of statements or batches that were active during the selected time frame. • This feature lets you check whether it is possible to improve performance by creating one or several indexes. • Once statistics are collected, remember that statistics should be periodically maintained.

Table Schema Change May Increase Its Accessing Time

The average In MS-SQL time increased after an object change. Check the changes and how they affected the statement execution time.

Table 7 Table Schema Change May Increase Its Accessing Time findings

Column	Description
What to do next	In the SQL tab, examine the information displayed in the History tab to identify object changes and correlate these changes to performance degradation.
Advice	Several changes may effect In MS-SQL time and may effect other statements In MS-SQL as well. Changes such as index creation/drop can cause access plan changes that can effect statement performance. Do the following: • Identify the change in the History tab • Compare statement or batch execution plans in the Compare tab.

Statement Is Not Scalable

Statement resource consumption was increased by n% as a result of an increase in its executions.

Table 8 Statement Is Not Scalable findings

Column	Description
What to do next	Perform one of the following options: • Examine scalability by analyzing degradation in average duration for statement resulting from an increase in the number of executions. • Check the program executing the statement and try to identify application changes. • Explore other statement or batch execution identifiers, such as, machine or users, in the Activity tab.
Advice	Perform the following options: • Try to determine if this is a pure scalability issue. Increase the time frame and explore statement scalability over a larger period of time. If it is a problem of scalability, this can be treated in many ways. • For example, you can avoid unscalable operations, such as table scans, in an execution plan. • Eliminate concurrency by distributing the operation of the application over several MS-SQL servers.

Table Growth May Increase Its Accessing Time

The average In MS-SQL time of the statement increased following a major change in table size.

Table 9 Table Growth May Increase Its Accessing Time findings

Column	Description
What to do next	Examine the information displayed in the History tab in the SQL tab to identify volume changes and correlate these changes to performance degradation.
Advice	Volume changes may effect response time especially in: • **Table scans**. May effect costs dramatically and create larger scans. • **Index scans**. May create larger scans. • **Index structure**. Every level in the index depth requires I/O synchronization. An index depth of four will result in four synchronized I/Os for each key, meaning that I/O cost will be too high and the optimizer may choose not to use the index. Try defragmenting the index or redesigning it. • Examine changes in the History tab and analyze these changes over time by launching to the Objects tab.

Increase in Resource Consumption

The SQL Server resources consumed by the statement increased by *n*%.

Table 10 Increase in Resource Consumption findings

Column	Description
What to do next	Perform one of the following options: • Examine the history of the statement. • Examine the index and statistics recommendations for the selected statement. • Examine the statement plan in the SQL tab. • Examine statement activity in the Activity tab.
Advice	The average In MS-SQL time of the statement or batch has increased but not as a result of major table growth, scalability change or schema changes. A major change can result from a change in the execution plan. Examine the history of the statement and try to identify whether a major change took place. Compare explain plans using the Compare tab. Check instance behavior. An increase in instance execution over time, may affect the performance of regular statements.

Statement or Batch Was Locked

Much of the statement or batch time was spent waiting for a lock. Regular locks can be categorized as follows:

- During the blocker session, a locked statement or batch ran for a short period of time. Afterwards the session was idle or continued running other statements or batches. In this case, it is possible to identify the blocker session, but not necessarily the blocker statement or batch.
- During the blocker session, a locked statement or batch ran for a long period of time. Identifying the blocker statement or batch is easier in this case.

Table 11 Statement or Batch Was Locked findings

Column	Description
What to do next	Perform one of the following options: • Check the statement or batch locking information and associated blocker sessions in the Activity tab. • Check the locked object in the Activity tab. • Check statement or batch activity in the Activity tab. Increase the time frame, if necessary.
Advice	Identify the blocker session in the Activity tab and examine the application and application timing. Examine the lock chain to identify the statement or batch holding the lock: • In the Activity tab sort statements or batches by their lock wait. • Identify major statements or batches. • Narrow the time frame. Drill down into major statements or batches. • In the Association area, select Blocker Sessions. • Try to identify lock patterns within a larger time frame.

Statement Activity Consistently High

Total In MS-SQL time of the statement was consistently high and reached the thresholds of the top statements.

Table 12 Statement Activity Consistently High findings

Column	Description
What to do next	Perform one of the following solutions: • Examine statement activity in the Activity tab. • Examine the index and statistics recommendations for the selected statement. • Examine the plan of the statement in the SQL tab.
Advice	The In MS-SQL time of the statement is consistently high for the select time frame, as compared with resource consumption of the previous week. This finding is issued when the statement shows up within the top resource consumers of your application, their resource consumption was always high, and no major change in resource consumption occurred during the last two weeks. Check which resource is most-consumed by the statement or batch and determine how you can make it available to the statement. For example: Consider scheduling different activities that uses the same resource at different times, thereby freeing the resources for the statement.

About Object findings

The following object findings can help you tune your system:

- Heavily Accessed Object
- Heavy Operators
- Missing Indexes
- Missing Statistics
- Table Schema Change May Increase Its Accessing Time
- Object Is Not Scalable
- Table Growth May Increase Its Accessing Time
- Increase in Resource Consumption
- Locked Object
- Index Overhead
- High Amount of Index Scans
- High Amount of Table Lookups
- High Amount of Unused Heap Pages

Heavily Accessed Object

Object is a major consumer of MS-SQL resources. By tuning the object, you can free resources needed by other statements and processes.

Table 13 Heavily accessed object findings

Column	Description
What to do next	Perform one of the following options: • Examine the SmarTune object findings for the selected object. • Examine the index and statistics recommendations in the Recommend tab in the Objects tab. • Examine table details in the Objects tab. • Examine table activity in the Activity tab.

Advice	The following scenarios indicate which factors can lead to heavy resource consumption and what steps you can take: • In MS-SQL time of the object increased. Identify and try to find the source of the change. An example of such a change can be table growth or statistics changes. • A missing index was identified (SQL Server 2005 only). The SQL Server optimizer identified missing indexes when the access plan was created. Detailed index information can be found in the Recommend tab in the Objects tab. • Locked Object. Try to identify blocker sessions. • The object is suffering from high number of index scans. Try to identify where a matching level problem exists, indicating that the index column order does not match the statement or batch WHERE columns. Example: An index on A, B, C, while the statement or batch is "select … where A=3 and C=7". Many irrelevant index leaf pages will be read in this case. This can be avoided by either changing the index column sequence or adding columns to the index. Analyze the recommendations provided in the Recommend tab in the SQL tab. • The object is suffering from high number of table lookups. Verify that all the columns in the select list are indeed needed. Try to use the INCLUDE (2005 only) option to add the needed columns to the index leafs to prevent unnecessary table lookups or in case of 2000 instance, use the index covering technique to eliminate extra table lookups

Heavy Operators

One or more statements access this object using heavy operators. Statement has major access plan operators for this object that scan a great deal of data and show a high cost percentage.

Table 14 Heavy Operators findings

Column	Description
What to do next	Perform one of the following options: • Examine the index and statistics recommendations for this object in the Recommend tab, in the Objects tab. • Explore table details in the Objects tab. • Examine table activity in the Activity tab.

Advice	One of the following operations (for example, table scan, index scan, clustered index scan or filter) was identified as a major cost indicator. Try to eliminate the major consuming operation by doing one of the following: For a table scan: • Eliminate a full table scan or cluster index scan. • Try to identify high selectivity columns for the statements or batches. Create an index matching the statement or batch's predicates. • Try to identify if there is a small set of columns that can be used in an index-only operation. • Partition the table according to the best predicates existing in the statements or batches. • Improve full table scan. Table fragmentation can affect overhead when a table scan or partial table scan is performed. Determine whether the table is heavily fragmented. If it is, defragment the table. For an index scan: • If a missing index finding is detected, launch to the Recommend tab. Otherwise try to identify where the matching level problem exists, indicating that the index column order does not match the statement or batch WHERE columns. Example: ◦ An index on A, B, C, while the statement or batch is "select … where A=3 and C=7". ◦ Many irrelevant index leaf pages will be read in this case. This can be avoided by either changing the index column sequence or adding columns to the index. ◦ Analyze the recommendations provided in the Recommend tab in the SQL tab. For a Filter: • Eliminate a filter by doing one of the following: ◦ Consider creating an index view. ◦ Consider performing an index on the computed column. For a Sort: • Try reducing the space or memory required for Sort operations by reducing the number of sorted columns or by filtering the rows to be sorted. • Try to find a way to eliminate the sort by using pre-sorted information like creating an index on the sorted columns. • Check the values of the *Minimum memory per query (KB)* and *Minimum memory for index create sorts (KB)* instance parameters. Use the *Memory for index create sorts (KB)* configuration to control the amount of memory used by index creation sorts. The *Minimum memory for index create sorts (KB)* configuration is self-configuring and should work in most cases without requiring adjustment. However, if you experience difficulties creating indexes, consider increasing the value of this option from its run value. Query sorts are controlled through the *Minimum memory per query (KB)*. For a Hash: • Hash join is used where the joined tables or row sets are large or where there aren't adequate indexes or the lack of indexes at all. ◦ Verify that there are no missing indexes or the indexes are adequate. ◦ Search for a missing WHERE clause or a missing condition in the WHERE clause. ◦ Search for a non-sargeable expression. A non-sargeable expression is an expression preventing the optimizer of using the index in the ideal way or not at all like a function expression. ◦ Check the values of the *Minimum Size of Server Memory (MB)* and *Max Size of Server Memory (MB)* instance parameters. For a Index Spool, Table Spool and Row Count Spool: • Spooling are internally temporary tables created by SQL Server on the tempdb database. Spooling may cause tempdb overhead and thus resulting in instance performance degradation caused by tempdb wait or tempdb major growth. ◦ Try to eliminate spooling operations by rephrasing the query. ◦ Try to reduce the number of logical reads or writes.

Missing Indexes

During SQL Server optimization, missing indexes were identified for one or more statements. This means that the SQL Server optimizer recommended creating indexes to improve the performance of the statement. Detailed information regarding index recommendation for this statement can be found in the Recommend tab of the Objects tab.

Table 15 Missing Indexes findings

Column	Description
What to do next	Examine the information displayed in the Recommend tab in the Objects tab to identify index recommendations.
Advice	Perform one of the following options: • Examine the information displayed in the Recommend tab in the Objects tab to identify index recommendations. • Use the What if feature in the Recommend tab to evaluate which statements were affected, based on a large set of statements that were active during the selected time frame. This feature lets you check whether it is possible to improve performance by creating one or several indexes.

Missing Statistics

The Missing Statistics warning was issued by the SQL Server optimizer when the access plan was created. This means that the SQL Server optimizer recommends creating and updating the statistics on the object. Detailed information on missing statistics for this object can be found in the Recommend tab in the Object tab.

Table 16 Missing Statistics findings

Column	Description
What to do next	Examine the information displayed in the Recommend tab in the Objects tab to identify missing statistics and analyze index recommendations.
Advice	Perform the following options: • Examine the information displayed in the Recommend tab in the Objects tab to identify missing statistics and analyze index recommendations. • Use the What if feature in the Recommend tab to evaluate a missing index, based upon a large set of statements or batches that were active during the selected time frame. This feature lets you check whether it is possible to improve performance by creating one or several indexes. Once statistics are collected, remember that statistics should be periodically maintained.

Table Schema Change May Increase Its Accessing Time

The total In MS-SQL time of the object increased after changes were made to the schema.

Table 17 Table Schema Change May Increase Its Accessing Time findings

Column	Description
What to do next	Examine table details and schema changes in the Objects tab.
Advice	Several changes may effect in MS-SQL response time and may effect other statement changes as well. Changes such as index creation /drop can affect statement performance: • Identify the change in the Objects tab. • Examine the change, when it was made, the performance changes related to the change, and identify any other changes in the object that may result from the change.

Object Is Not Scalable

The total In MS-SQL contribution of the object increased when the number of statements executions changed.

Table 18 Object Is Not Scalable findings

Column	Description
What to do next	Examine the object’s behavior over time and explore the usage patterns of the statements.
Advice	Perform one of the following options: • Examine the object’s behavior over time and explore the usage patterns of the statements. • Check whether usage patterns dramatically changed. If yes, consider improving them by indexing or partitioning. • Identify major users by associating to users or machines in the Association. Determine if this is acceptable for your system based on your knowledge of your system.

Table Growth May Increase Its Accessing Time

The total In MS-SQL time of the object increased following a major change in the table size.

Table 19 Table Growth May Increase Its Accessing Time findings

Column	Description
What to do next	Examine table details and associated statements in the Objects tab to identify statements that may be affected by table growth.

Advice	Volume changes may effect response time especially in: • Table scans: may effect costs dramatically. • Index structure: Every level in the index depth requires I/O synchronization. An index depth of four will result in four synchronized I/Os for each key, meaning that I/O cost will be too high and the optimizer may choose not to use the index. Try defragmenting the index or redesigning it. Try to identify the heaviest statements and tune them.

Increase in Resource Consumption

The total In MS-SQL time of the object has increased.

Table 20 Increase in Resource Consumption findings

Column	Description
What to do next	Perform one of the following options: • Examine table details of common access patterns and associated statements in the Objects tab (SQL Server 2005 only). • Examine the recommendations for the index and statistics of the selected object. Explore table details in the Objects tab.
Advice	The average In MS-SQL time of the object has increased but not as a result of major table growth, scalability change or schema changes. A major change can result from a change in execution plan. Examine the table details and the statement accessing it and try to identify what caused the change. Compare explain plans using the Compare tab.

Locked Object

Much of the In MS-SQL time was spent waiting for a lock on the table.

Regular locks can be categorized as follows:

- During the blocker session, a locked statement ran for a short period of time. Afterwards the session was idle or continued running other statements. In this case, it is possible to identify the blocker session, but not necessarily the blocker statement.
- During the blocker session, a locked statement ran for a long period of time. Identifying the blocker statement is easier in this case.

Table 21 Locked Object findings

Column	Description
What to do next	Perform one of the following options: • Examine locking information in the Activity tab. • Examine the index and statistics recommendations for the selected object. • Explore table details in the Objects tab.
Advice	Identify the blocker session in the Activity tab and examine the application and application timing. Examine the lock chain to identify the statement holding the lock.

Index Overhead

Most of the activity on the index is due to the fetching of index pages from the disk, reflecting changes made by INSERT, DELETE, and UPDATE statements.

Table 22 Index Overhead findings

Column	Description
What to do next	Perform one of the following options: • Examine the statements causing index updates, in the Statements tab. • Check if the index is being used in execution plans. • Try to identify index update patterns (such as, daily or nightly), in the In Oracle graph, in the Read/Write Operations tab.

Advice	Perform one of the following options: • When insert statements are part of a load, batch, or night activity, consider dropping the index before performing the activity, and recreating it afterwards. • If the index is not used in execution plans consider dropping the index or unused columns from the index, to reduce index overhead. If the index is used in execution plans, launch to the What-If tab to see which statements may be effected by this change.

High Amount of Index Scans

The index is mainly used for scans and not for seeks which is the recommended method.

Table 23 High Amount of Index Scan findings

Column	Description
What to do next	• Examine the information displayed in the 'Common Access Patterns' and Operational Statistics' • Associate to statements • Examine the information displayed in the Recommend tab in the SQL tab to identify missing statistics and analyze index recommendations.
Advice	Perform one of the following options: • Examine the index scans percentage of overall usage of the index • Associate to statements and find out why the optimizer chose to use index scans by looking for missing predicates, functions on the index columns, non-selective columns, range scan queries, inadequate indexes, etc. • Examine the information displayed in the Recommend tab in the SQL tab to identify missing statistics and analyze index recommendations. • Use the What If feature in the Recommend tab to evaluate which statements or batches were affected, based on a large set of statements or batches that were active during the selected time frame. This feature lets you check whether it is possible to improve performance by creating one or several indexes.

High Amount of Table Lookups

The table has a high amount of lookups which can cause excessive I/O wait.

Table 24 High Amount of Table Lookup findings

Column	Description
What to do next	Examine the information displayed in the Recommend tab in the SQL tab to identify and analyze index covering recommendations.
Advice	Perform one of the following options: • Examine the information displayed in the Recommend tab in the SQL tab and check for index covering recommendations. • Use the What If feature in the Recommend tab to evaluate which statements or batches were affected, based on a large set of statements or batches that were active during the selected time frame. This feature lets you check whether it is possible to improve performance by creating one or several indexes.

High Amount of Unused Heap Pages

SQL Server reads a high amount of unused Heap Pages, which can lead to unnecessary I/O Wait.

Table 25 High amount of unused Heap Pages

Column	Description
What to do next	Examine the difference between the number of rows and allocated pages and consider rearranging the table by creating a cluster index and then dropping it.

About Instance findings

The following instance findings can help you tune your system:

- Locked Instance
- Tempdb Bottleneck
- Buffer Cache Is Too Small

- Other Applications Influence SQL Server (Memory)
- Other Applications Influence SQL Server (CPU)
- Transaction Log Bottleneck
- Extensive Internal Wait
- Tempdb Major Growth
- High CPU Wait

Locked Instance

x% of the In MS-SQL time was spent waiting for locks. Regular locks can be categorized as follows:

- During the blocker session, a locked statement or batch ran for a short period of time. Afterwards the session was idle or continued running other statements or batches. In this case, it is possible to identify the blocker session, but not necessarily the blocker statement or batch.
- During the blocker session, a locked statement or batch ran for a long period of time. Identifying the blocker statement or batch is easier in this case.

Table 26 Locked Instance findings

Column	Description
What to do next	Perform one of the following options: • Examine the table activity and locking information in the Activity tab • Examine lock counters in the Statistics tab.
Advice	To reduce the lock wait for the instance, consider the following solutions: Concentrate on locked statements: • In the Activity tab sort statements by their lock wait. • Identify a major statement. • Narrow down the time frame and drill down into it. • Change association to Blocker Sessions. • Concentrate on locked objects. Check to see if the lock appears in the Current tab. If yes, examine the lock chain to identify the statement holding the lock. • Concentrate on blocker sessions. Try to identify the locking statement in the Activity tab using a narrower time frame matching, the lock periods. Focus on the locked table and associated statements. The DML statement (and update queries) that are NOT waiting for locks can be the immediate suspects.

Tempdb Bottleneck

The instance is experiencing a bottleneck of the tempdb database. *x*% of the In MS SQL time was spent waiting for tempdb. The tempdb database is used for temporary storage for sorting, joining, and, in SQL Server 2005, for row versioning.

Table 27 Tempdb bottleneck findings

Column	Description
What to do next	• Examine the type of statements heavily using the tempdb resource in the Activity tab. • Check that a sort is being performed on the statements and tune the Sort operation. Tuning a Sort operation can be performed by making sure that only the relevant columns and rows are being sorted. You can view the columns being sorted in the SQL tab. • Check how the tempdb files are created in the Objects tab. For example, check on which devices the files reside and check the auto-extend parameters (a small value can create fragmentation at the OS level). • Check the load on the devices on which the tempdb files are created. You can view this in the Statistics tab.

Advice	• Tune sorts or joins:
	  ◦ Try reducing the space required for Sort operations by reducing the number of sorted columns or by filtering the rows to be sorted.
	  ◦ Verify that Hash and Merge joins occurred as the result of the proper join methodology. Hash and Merge joins are temporary storage consumers.
	  ◦ Create indexes to eliminate the sorts or change the join methodology to a Nested Loop.
	  ◦ Check the values of the `min server memory (MB)` and `index create memory (KB)` instance parameters.
	• Tune tempdb files:
	  ◦ Verify that tempdb files are distributed across several devices.
	  ◦ Check the load on each device.
	  ◦ Verify the auto-extend parameters.
	  ◦ Verify that the file is not fragmented at the OS level.
	• Tune row versioning:
	  ◦ Verify that row versioning is enabled only when needed.
	  ◦ Verify that the Snapshot Isolation level is used properly.
	• Examine tempdb usage when the applications explicitly create temporary tables. Try creating indexes on those tables after they are populated.

Buffer Cache Is Too Small

A SQL Server instance is configured to work with a small buffer cache. Your applications are experiencing *x*% of the In MS-SQL time for I/O wait. The application's overall performance is therefore not optimal.

Table 28 Buffer Cache Is Too Small findings

Column	Description
What to do next	• Examine the hit ratio of the caches in SQL Server, in the Statistics tab. • Examine the page faults the SQL Server issues. A small number of page faults and a low hit ratio can indicate that the SQL Server is not maximizing the use of server memory. • Examine the memory being allocated to the SQL Server as opposed to the memory available to the server. • Examine the memory consumption of other processes in the Insight Savvy for OS. • Examine how long the applications wait for I/O. • Examine page life expectancy, in the Statistics tab.
Advice	Before increasing memory for the SQL Server, check that you are not going to adversely affect other applications running on the server by doing so. In addition, adding increasing the SQL Server memory too much, can decrease the performance, because SQL Server will have to do paging. Experts recommend allowing operating systems to have 20% available memory. If you have other applications running on the same server, check their memory requirements in the Insight Savvy for OS. When you determine how much memory should be allocated to SQL Server, you can change its settings. In this case you can configure memory to be fixed and not dynamic. Check the value of the min server memory (MB) instance parameter. This parameter indicates how much memory is allocated to every session connected to the SQL Server. If you find that you are experiencing many idle sessions (due to the threshold settings in the connection pooling) you are wasting too much memory. SQL Server does not allow the configuration of different settings for each cache, so you must verify whether or not the buffer cache has enough memory.

Other Applications Influence SQL Server (Memory)

A SQL Server instance experienced many page faults and lack of memory, while other processes on the same server did not experience any page faults problem.

Table 29 Other applications influence SQL Server (memory) findings

Column	Description
What to do next	• Examine the SQL Server page faults, in the Statistics tab. • Examine the memory allocated to SQL Server. • Examine the memory resources used by other processes on the server. • If Insight Savvy for OS is installed, check the memory usage of all processes on the server.
Advice	Because SQL Server is your RDBMS, and it is the single resource being used by most of your applications, it is essential that it receive all the resources it requires. You should be aware of which applications are also running on the server and how they affect your SQL Server. Examine the amount of memory being used by the SQL Server and the amount of memory being used by the other processes located on the server. Verify that SQL Server has enough memory allocated to it. Prioritize the other applications running on the server and decide if any of the other processes can be migrated to another server, thereby freeing memory resources to the SQL Server.

Other Applications Influence SQL Server (CPU)

A SQL Server instance experienced CPU shortage, as a result of other processes running on the server.

Table 30 Other applications influence SQL Server (CPU) resources findings

Column	Description
What to do next	• Examine CPU usage of the server, in the Statistics tab. • Examine In MS-SQL breakdown, in the Activity tab. • If Insight Savvy for OS is installed, examine CPU usage of all processes on the server.

Advice	Because SQL Server is your RDBMS, and it is the single resource being used by most of your applications, it is essential that it receive all the resources it requires. You should be aware of which applications are also running on the server and how they affect your SQL Server. Examine the amount of CPU being used by the SQL Server and the amount of memory being used by the other processes located on the server. Verify that SQL Server has enough CPU allocated to it. Prioritize the other applications running on the server and decide if any of the other processes can be migrated to another server, thereby freeing CPU resources to the SQL Server.

Transaction Log Bottleneck

Some of the databases suffer from transaction log bottleneck. Log wait consumes x% of the In MS-SQL time.

Table 31 Transaction Log Bottleneck findings

Column	Description
What to do next	• Examine which database in the instance suffers from transaction log bottleneck, in the Activity tab. • Examine the In MS-SQL breakdown of the applications, in the Activity tab. • Examine the load on the devices holding the transaction logs, in the Statistics tab. • Examine Log Flush counters, in the Statistics tab.
Advice	The transaction log is the file that experiences many writes and few reads. Every update to the data is eventually written to the transaction log. Since every database has its own transaction log, the first thing to do is to try to pinpoint the specific database. If all of databases are experiencing a transaction log bottleneck, try examining the recovery interval (min) configuration. When you have identified which database is experiencing a transaction log bottleneck, examine the applications using this database. A bottleneck in the transaction log indicates that your application performs commits at a very high or very low frequency. This means that many records must be saved at the same time. The challenge is finding how many commits to do and how many records to save in each commit. You can also consider moving the transaction logs to different and faster devices.

Extensive Internal Wait

The SQL Server instance has spent much of its time waiting for Internal Wait.

Table 32 Extensive Internal Wait findings

Column	Description
What to do next	• Examine which database in the instance suffers from Internal Waits, in the Activity tab. • Examine the Internal Wait category using the Internal Waits view, in the Activity tab. • Examine a specific Internal Wait in the Statistics tab (SQL Server 2005 only).

Advice	An Internal Wait is divided into the following substates:
	
	• **Buffer Pool**. Groups together the events that show contention on pages in the buffer pool. Buffer wait on tempdb pages are considered to be part of the tempdb wait.
	• **Latch**. The session is waiting for an internal lock to be released.
	• **Parallel**. The session is waiting for one of its sub-threads to complete its operation.
	• **DTC**. Aggregates waits that occur when Distributed Transaction Coordinator (DTC) sessions have to wait for each other. *This state is only available for SQL Server 2005 instances.*
	• **DB Mirror**. Aggregates the new waits that occur when DB mirroring is performed, such as the waits that occur if the communication layer used by DB mirroring becomes backlogged. *This state is only available for SQL Server 2005 instances.*
	• **Profiler**. Aggregates a number of states associated with the SQL Profiler and lets you see how much of the database resources it consumes. *This state is only available for SQL Server 2005 instances.*
	• **Memory**. Aggregates several types of waits that indicate that a session is waiting for memory to be allocated to it. This state is only available for SQL Server 2005 instances.
	• **Backup**. Includes the wait type that commonly occurs when a Backup command is performed. This state is only available for SQL Server 2005 instances.
	• **Other Internal waits**. Aggregates the following types of waits:
	  ◦ **Full text waits**. Includes wait types dedicated to the full text indexing service and appears whenever a full text index is in progress.
	  ◦ **HTTP waits**. Includes waits that occur when HTTP and SOAP operations are executing.
	• **Query notifications**. Aggregates a number of states associated with the synchronization of Query Notification sessions. *This state is only available for SQL Server 2005 instances.*

Tempdb Major Growth

The tempdb database has experienced major growth. The tempdb database is used for temporary storage for sorting, joining, and, in SQL Server 2005, for row versioning.

Table 33 Tempdb Major Growth findings

Column	Description

What to do next	Perform one of the following options:
	
	• Examine the current tempdb content to find out which resource type is being consumed the most:
	  ◦ User temporary tables
	  ◦ Internal objects such as temporary system tables caused by Sort, Hash Join or Spooling operations.
	• Examine the type of statements heavily using the tempdb resource in the Statements tab.
	• Examine the Statistics to find out the number of files/tables creation number and rate.
	• Examine the Tempdb Usage to find out open transactions preventing tempdb to shrink.

Advice	• Examine the type of statements heavily using the tempdb resource in the Statements tab.
	  ◦ Tune sorts.
	  ◦ Try reducing the space or memory required for Sort operations by reducing the number of sorted columns or by filtering the rows to be sorted.
	  ◦ Hash and Merge joins are temporary storage consumers. Try to find a way to eliminate the sort by using pre-sorted information like creating an index on the sorted columns.
	  ◦ Check the values of the min memory (MB) and index create memory (KB) instance parameters. Use the index create memory option to control the amount of memory used by index creation sorts. The index create memory option is self-configuring and should work in most cases without requiring adjustment. However, if you experience difficulties creating indexes, consider increasing the value of this option from its run value. Query sorts are controlled through the min memory per query option.
	• Tune Hash Joins:
	  ◦ Verify that there are no missing indexes.
	  ◦ Verify that there are adequate indexes.
	  ◦ Search for a missing WHERE clause.
	  ◦ Search for a non-sargeable expression. A non-sargeable expression is an expression preventing the optimizer of using the index in the ideal way or not at all like a function expression.
	• Tune Index Spool, Table Spool and Row Count Spool:
	  ◦ Spooling is internally temporary tables created by SQL Server on the tempdb database. Spooling may cause tempdb overhead and thus resulting in instance performance degradation caused by tempdb wait or tempdb major growth.
	  ◦ Try to eliminate spooling operations by rephrasing the query.
	  ◦ Try to reduce the number of logical reads or writes.
	• Tune row versioning. If a version store is not shrinking, it implies that a long-running transaction is preventing version store cleanup.
	• Examine tempdb usage when the applications explicitly create temporary tables. Try creating indexes on those tables after they are populated.
	• Examine the Tempdb Usage to find out open transactions preventing tempdb to shrink.

High CPU Wait

The instance used SQL Server resources while waiting for CPU.

Table 34 High CPU Wait

Column	Description
What to do next	• Examine the high CPU usage statements. • Examine the CPU usage to find the number of waiting for CPU tasks. • Examine where the schedulers are not evenly loaded on the instance. • Examine the overall CPU usage (using Insight OS) to find out if other processes occupy the CPUs.

About Storage findings

The following Storage findings can help you tune your system:

- Storage Contention on Device (Clariion)
- Storage Contention on Device (Symmetrix Thick)
- Storage Contention on Device (Symmetrix Thin)
- Storage Contention on Device (Symmetrix F.A.S.T. VP)
- Storage Contention between Log and Datafiles
- Unbalanced Storage Devices Activity
- Storage with Low Cache Hit Ratio
- Storage Contention on tempdb

Storage Contention on Device (Clariion)

The instance spent *x*% of its resources waiting for I/O on the specified storage device.

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 35 Storage Contention on Device findings

Column	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage contention in Physical Disks, Devices and Files on the same raid group. • Examine disks statistics.
Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Device (Symmetrix Thick)

The instance spent *x*% of its resources waiting for I/O on the specified storage device.

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 36 Storage Contention on Device findings

Column	Description

What to do next	• Examine the device activity over time and database files contention. • Examine storage contention in Physical Disks, Devices and Files on the same raid group. • Examine disks statistics.
Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Device (Symmetrix Thin)

The instance spent *x*% of its resources waiting for I/O on the specified storage device.

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 37 Storage Contention on Device findings

Column	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage contention in Physical Disks, Devices and Files on the same raid group. • Examine disks statistics.
Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention on Device (Symmetrix F.A.S.T. VP)

The instance spent *x*% of its resources waiting for I/O on the specified storage device.

The fact that a storage device (LUN) is causing a lot of I/O waits could be caused from an intensive load or as a result of two sorts of contentions: a logical contention (e.g. imbalanced activity of the database) or a physical contention (e.g. one of the underlying physical devices is being shared with another heavy I/O consuming activity).

Table 38 Storage Contention on Device findings

Column	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage contention in Physical Disks, Devices and Files on the same raid group. • Examine disks statistics.

Advice	• If the device is loaded by the monitored database only and by a singular entity (e.g. a file, object, or partition), consider splitting this load (e.g. separating the objects in the file, partitioning the object, etc). • To relieve inter application logical contention, check if the database's I/O activity is balanced. Spread heavy I/O consuming files across the storage devices, to avoid a situation in which few heavy files reside on the same storage device. • To relieve intra application logical contention, check whether there are additional applications using the storage device. For example, if the number of I/O requests processed by the storage device is significantly higher that the requests sent by the database, it means that the storage device is being used by an additional application. • To relieve physical contention, check whether there is significant I/O activity in the underlying shared physical disks and raid group. Another potential cause of contention are the EMC adapters (front director and disk director). If the load is imbalanced, consult with the storage administrator about relocating the information to other disks which reside on a more vacant location. • Consider storage tiering - a faster device may reduce the I/O wait time significantly.

Storage Contention between Log and Datafiles

The instance spent *x*% of its resources waiting for I/O on the specified storage device. The device contains both Log files and Datafiles.

Transaction Log files are frequently accessed by the database. The majority of the operations performed are writing commands, which cause a heavy load on the underlying disks.

As these files are considered heavy I/O consumers, it is highly recommended to place them on a separate disk without other database files. Separating the Transaction Log files by placing them on different volumes (e.g. E:/ and F:/) may not be enough, as the storage devices (LUNs) and physical disks may be shared between several file systems and volumes.

Table 39 Storage Contention between Log and Datafiles findings

Column	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage contention in Physical Disks, Devices and Files on the same raid group. • Examine disks statistics.
Advice	It has been detected that the Transaction Log files share the storage devices (LUNs) with other database files. Consult the storage administrator about provisioning the storage devices (LUNs) better to avoid this.

Unbalanced Storage Devices Activity

The instance I/O activity is not balanced across storage devices.

There are several storage devices (LUNs) allocated to the instance. However, the I/O activity is not spread evenly across these storage devices. The contention on the heavy storage devices increases the response time for the activities run on them. Such a situation can be caused by imbalanced internal database activity, contention on the storage device by other applications or an inefficient RAID policy.

Table 40 Unbalanced Storage Device Activity findings

Column	Description
What to do next	• Compare the storage devices activity over time. • Examine the storage devices statistics.
Advice	• In the activity tab, check which database files are the most I/O consuming and spread them evenly across the storage devices. • Consult with the storage administrator and check for other applications using the same storage devices or their underlying physical disks. • Consult with the storage administrator about the RAID policy. A different striping may spread the I/O load across the storage devices.

Storage with Low Cache Hit Ratio

The instance was waiting for I/O on storage devices with low cache read hit ratio.

I/O requests sent to the EMC storage array are processed on the storage internal cache. EMC runs a "prefetch" mechanism that predicts the future blocks to be requested and load them to the cache. Requests that are not being served by the cache access the underlying disks, and are significantly slower. The cache can be shared by different storage devices (LUNs), and may potentially become a layer of contention between I/O activities which are being served by those storage devices.

Table 41 Storage with Low Cache Hit Ratio findings

Column	Description
What to do next	Examine the storage devices statistics and hit radio.
Advice	• It has been detected that the storage devices (LUNs) that serve the instance get bad cache performance (i.e. low "hit ratio"). To relief the cache contention, consult the storage administrator about the following: • Expanding the storage cache allocated to the database devices. • Enabling "EMC Cache Partitioning", to isolate the instance cache and avoid external contentions.

Storage Contention on tempdb

Tempdb datafiles are frequently accessed by the instance. The majority of the operations performed are writing commands, which cause a heavy load on the underlying disks.

As these files are considered heavy I/O consumers, it is highly recommended to place them on a separate disk without other database files. Separating the tempdb datafiles by placing them on different volumes (e.g. E:/ and F:/) may not be enough, as the storage devices (LUNs) and physical disks may be shared between several file systems and volumes.

Table 42 Storage Contention on tempdb findings

Column	Description
What to do next	• Examine the device activity over time and database files contention. • Examine storage contention in Physical Disks, Devices and Files on the same raid group. • Examine disks statistics.
Advice	It has been detected that the undo tablespace files share the storage devices (LUNs) with other database files. Consult the storage administrator about provisioning the storage devices (LUNs) better to avoid this.