Usage concepts

This section includes the following topics:

How information is collected

Precise for SQL Server collects and displays performance data that enables quick and accurate performance monitoring and analysis. To understand the information displayed in the various graphs, tables and views, it is important to understand how information is collected.

About duration values

Precise for SQL Server displays duration values for several entities. The following table describes the duration value for each type of entity.

Table 1 Duration values for entities

Entity	Description
Current session	The time that has elapsed since the session started (login time). This is not the time the session was active executing statements. For example, if a session was connected to SQL Server for one hour without executing any statement, the duration will still be one hour.
Programs (such as, Logins, Machines, DB Users, Databases, time units)	The elapsed time of the sessions executing the program. ⓘ The time is not measured from the login time (as it is in the Current session entity) but from the moment the Collector started. The average duration counter displays the average time each program has been running.
Statement (batch)	The time that elapsed while the statement was executed. This is very similar to the duration of programs (see above); but since the statement is almost always active, this duration also indicates the time the statement was active. The average duration counter displays the average execution time of the statement.

If Interpoint is installed (for SAP, PeopleSoft, Siebel or COM+), the duration and average duration fields have no significant meaning and can be ignored.

About session states

SQL Server provides reporting on many different types of events. Precise for SQL Server groups events according to logical relationships to enable you to easily pinpoint heavy resource consumers.

For example, when the Collector samples the SQL Server, it assigns a state for each connection. An application may wait for many events in SQL Server. To simplify analysis, and help you to easily identify the source of resource consumption, Precise for SQL Server groups various wait types into wait groups or states and defines the status of sessions currently connected to SQL Server by their states. The state is calculated from the following sysprocess columns: waittype, waittime, and status, and is divided into two types of states: In MS-SQL and Non-In MS-SQL.

Precise for SQL Server also provides statistics about the waits encountered by threads that are in execution. This information is only available for SQL Server 2005 instances. Diagnosing these counters can provide important information on performance issues throughout the system.

About In MS-SQL states

The In MS-SQL session states signify that the session is performing activities in the SQL Server. The following table describes In MS-SQL session states.

Table 2 In MS-SQL session states

Icon	In MS-SQL State	Description
	Using CPU	The session is currently executing and not waiting for any other event. Using CPU can indicate a performance issue if the value exceeds 90% of the In MS-SQL value. When SQL Server does read ahead, the session does not suffer from I/O wait, but may still show a great deal of CPU usage.

	I/O Wait	The session is waiting for an I/O operation to be completed or terminated. I/O operation is much slower than CPU operation so it is very important to monitor whether an application is suffering from I/O wait. I/O can be easily resolved by reorganizing files across disks or by adding new disks to your system.
	Tempdb Wait	The session is waiting for an I/O operation or a buffer wait on tempdb pages. A high value may indicate a bottleneck in tempdb. Tempdb is the most heavily shared resource across the entire instance and can affect the performance of all applications. It is therefore important to monitor its usage. SQL Server 2005 uses tempdb for row versioning and online index builds, in addition to managing temporary objects, sorting and hashing.
	Lock Wait	The session is waiting for a lock to be released. The lock is held by another session.
	Redo Log Buffer Wait	The session is waiting for an operation of the log file to terminate. This state is generally encountered during a commit or rollback operation. A high log wait value may indicate a problem with the commit frequency in the processes experiencing massive updates to data.
	Memory Wait	The session is waiting for an internal resource to be freed. This state is divided into substates.
	Remote Wait	The session is waiting for a remote query to terminate. Some of the DBCC commands that check the database pages also create this type of wait. A remote resource can be either SQL Server and Oracle.
	CLR Wait	The session is currently performing a common language runtime (CLR) execution and is waiting for another CLR task or event to be initiated or completed. All CLR-related waits are grouped together under this category. This state is only available for SQL Server 2005 instances.
	Buffer Wait	The session is completing a network I/O operation. A high value can occur if results from a large result set are transferred to the client using shared memory netlibs or TCP/IP. In this case it is important to verify that only required rows and columns are returned.

About internal waits

The Internal Waits view displays internal waits, indicating that the session is waiting for an internal resource to be freed. An Internal Wait is divided into substates.

The following table describes the internal wait substates reported by Precise for SQL Server.

Table 3 Internal wait states

Internal State	Description
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 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.

About lock waits

The Locking view displays a breakdown of lock waits, indicating that the session is waiting for an application lock to be freed. Lock waits are divided into substates.

Table 4 Lock wait states

Lock Wait	Description
Row lock	The lock is being held on a row in a heap table.
Key lock	The lock is being held on a key in an index or a row in a clustered index.
Page lock	The lock is being held on a page.
Table lock	The lock is being held on the entire table.
Other lock	Includes locks on database, extents and files.
MD statistics lock	Indicates a metadata lock is being held on the statistics of a table or index.
MD partition lock	Indicates a metadata lock is being held on a partition function or partition schema.
MD other lock	Indicates a general metadata lock is being held.

About Non-In MS-SQL states

The Non-In MS-SQL session states signify activities performed outside of the SQL Server. The following table describes Non-In MS-SQL session states.

Table 5 Non-In MS-SQL session states

Icon	State	Description
	Request Wait	The session is waiting for the client to issue a request. A high value can indicate that a connection pool is too large and that many connections have been left open and are not active.
	Parallel Exchange Wait	The thread in a parallel session is waiting for data exchange from another thread.
	Idle	Indicates that the session is waiting for something other than a user request or is executing the WAITFOR DELAY command.

About Statement identifiers

Each statement is identified by the following parameters:

- **Text**. The actual statement text.
- **Database**. Each statement is associated with a single database.
- **User**. Each statement is associated with a single user (also called the parsing user).

For example, the statement "select * from sysobjects" executed in the master and msdb databases is represented by two different statements in the product. The ability to roll up the performance of those statements to one is also important. This is supplied using other identifiers such as the program entity.

How Precise for SQL Server monitors availability

The Precise for SQL Server Collector monitors SQL server every second and collects data regarding resource consumption. If the instance status (Up /Down) was changed, a message code is sent by the Collector and causes the current availability of the instance saved in the PMDB to be changed. In addition, the availability of the instance is saved, over time, in the PMDB. During a planned downtime period the instance availability is showed as it was available during this period.

If the Precise for SQL Server Collector is down, the instance availability status reported is "Unknown."

About application performance tuning

Precise for SQL Server allows you to proactively monitor, analyze, and tune SQL Server instances. By capturing, measuring, and correlating performance metrics from all critical system components. Precise for SQL Server provides a complete view of your application's performance. It detects business performance problems and helps you find suitable and comprehensive solutions thereby ensuring that your business applications always perform at peak efficiency.

Detailed information on the resource consumption of an application is extensive and includes data on CPU consumption, I/O waits, internal database waits, and so on. Still, with Precise for SQL Server you can easily identify the top resource consumers at different levels and at different times.

For example, you may want to start at the Instance level and then drill down to different Login names and the SQL statements they executed. You can view the status of current sessions and examine their consumption during the last minute, or focus your analysis on a particular time period, such as yesterday, last week, or last year. You can also examine changes in resource consumption patterns over time in different time units, such as:

- 15 minutes (time slice)
- hours
- days
- weeks
- months

About viewing current activity

The Current tab displays near real-time information on all sessions that are connected to your SQL Server instance during the last minute. Use the Current tab to solve bottlenecks, such as those caused by locked sessions or runaway processes. The Current tab displays activity information up to the last time slice of activity.

You can select an SQL Server instance and examine its resource consumption during the last time slice, such as CPU usage, I/O wait, and lock wait.

The application resource consumption is displayed on the instance level (as shown above) or as a summary of the session level of resource consumption (as shown below).

If you identified a blocking situation that is currently in progress (for example, there is a session waiting for a lock), select the Locks view to view the current sessions that are blocking the other sessions. The information is displayed in a tree format, which helps you easily identify which are the locked sessions, who they are being locked by and what is the lock level. You can drill down to the sessions involved in the lock to obtain more information about what is causing the lock. For example, you can drill down to a locking session to determine which SQL and which program are currently being executed.

About analyzing instance performance

While the Current tab displays near real-time data, it is recommended to begin the actual tuning process is the Activity tab.

The Activity tab allows you to analyze performance behavior over time or when the problem is reported (since in many cases the performance problem may no longer be evident from the data displayed in the Current tab). Using the Activity tab, you can analyze what happened at a specific point in time, identify problematic time periods, and drill down to analyze a bottleneck that caused a performance problem. You can begin by selecting an Instance and viewing the In MS-SQL overtime graph to identify problematic time periods.

In addition, long-term analysis is also required for understanding performance patterns and predicting future resource consumption. You can use the Activity tab to detect resource consumption patterns and suspicious deviations in past behavior.

The Activity tab can also be referred to when planning your system's capacity, performing period-to-period comparisons, and determining how best to optimize storage. Use the Activity tab to examine resource consumption of your SQL Server instance, over time, as well as the resource consumption of your databases, programs, logins, and so on. You can also identify dominant resource consumption by selecting the Overview view.

The Overview view provides a breakdown of the In MS-SQL states, such as, CPU usage, I/O waits, Lock waits, and Remote waits. Examining this data allows you to pinpoint the type of resource causing the bottleneck. If for example you identify that your SQL Server is suffering from an I/O wait problem, upgrading your server to a much more powerful machine will probably fail to provide you with the expected results.

A high CPU rate exceeding 90% may indicate that indexes are not being used.

About identifying resource consumers

Precise for SQL Server lets you drill down to additional levels to determine which entities are responsible for a particular performance problem. In the Association area you can observe different entities, such as databases, login names, users, programs, and client machines, to pinpoint the application component that consumes the most database resources.

For each entity (as, for example, the program entity), you can access the following information:

- Entities that are related to the entity (such as, the logins or machines that executed the program).
- Resources consumed by the entity. This enables you to determine if the entity is CPU or I/O bound, or waiting for a lock.

- Resource consumption patterns, which allows you to observe and identify trends in resource consumption.

You can drill down even further to other entities associated with the selected entity in the Association area. For example, once you've identified a problem in a specific program you can drill down to the batches or statements the program issued, to determine the source of the problem.

About correlating with application server transactions

Precise for SQL Server lets you correlate between SQL Server resource consumption and internal application server data, as follows:

- For SAP applications you can drill down to the most resource-consuming SAP transaction, SAP user, SAP work type, and SAP application server name.
- For COM+ applications you can observe information on the most resource-consuming COM+ application, COM+ component, COM+ method, and COM+ interface.
- For PeopleSoft applications you can identify the most resource-consuming PeopleSoft Client Machines, PeopleSoft User IDs, Tuxedo OS Users, Tuxedo domains, and work types.
- For Siebel application you can identify the most resource-consuming Siebel Application, Views, Batch Components Groups, Batch Servers Components, Business Objects, Siebel users and work types.

These entities are extensions to the common application performance entities. You may use the Association area to drill down and investigate the relationship between various entities.

About associating with offending SQL statements

Having identified a major resource-consuming entity, you can further drill down to the statements and batches executed by that entity. These may include long-running SQL statements or SQL statements that use few resources but are executed frequently.

For each statement or batch, you can access the following information in Precise for SQL Server:

- Entities that executed the statement or batch.
- Resources consumed by the statement or batch, which lets you determine if the statement or batch is CPU or I/O bound, or waiting for a lock, and so on.
- Resource consumption patterns that allow you to observe if the statement or batch execution is affected by excessive use of the same resource, by another statement or batch.

The drill down to most resource-consuming SQL statements or batches is also available from all COM+, SAP, PeopleSoft and Siebel extended entities.

About analyzing previous blocking situations

One of the main reasons that statements possess long execution times is that they are waiting for a resource that is locked by another session.

Using Precise for SQL Server, you can easily identify the following:

- Statements waiting for a locked object
- Locked objects (that is, contended objects)
- Sessions causing locks

In Precise for SQL Server, a statement waiting for the lock on a resource to be freed is in a Lock wait. The session holding the lock is called a Blocker session.

About SQL statement tuning

Most problems occur in SQL statements or batches. Precise for SQL Server lets you easily identify the most resource-consuming SQL statement or batch and view a detailed explanation of the execution plan of the SQL statement or batch. It can also help you resolve the problem by recommending which index would provide a better execution plan and reduce resource consumption.

If you are running SQL Server 2005 you can use the What-If option to check how creating a new index would influence the running of the application.

About identifying the most resource consuming statements or batches

Begin with the SmarTune tab to view a list of statements that require tuning. In the SmarTune tab, you can view expert knowledge regarding any area of your application showing performance deterioration and obtain comprehensive advice on how to solve specific problems. The Findings area lists the problematic statements while more comprehensive information is displayed in the Details area.

From the SmarTune tab you can launch, in-context, to specific displays in other tabs, to continue your analysis of the resource-consuming statement. The SmarTune tab provides guidelines that can help you narrow down your analysis and reach the root of a problem quicker and easier.

In the Association area of the Activity tab, you can associate to a specific database entity and then from there to its associated SQL statements.

When selecting an SQL statement, you can view a breakdown of its resource consumption over time, its text, and all locking contentions that it participated in (time spent waiting for a resource).

About collapsing constants into bind variables

When an application uses constant values and literals instead of bind variables (parameter markers), many different SQL statements actually constitute the same SQL statement with different values.

When each of these similar SQL statements is treated as a different statement, it is hard to summarize their resource consumption to identify the best candidate for SQL tuning.

Precise for SQL Server provides a unique algorithm that ignores constant values and groups all similar statements into collapsed statements. This enables you to identify the most resource-consuming collapsed statement.

The performance data for the collapsed statement is a summary of all performance data for all statements in the group. Most of the time, all the statements that were grouped together into one collapsed statement have the same execution plan. However, different values in the WHERE clause may lead the optimizer to choose a different execution plan.

Precise for SQL Server lets you select a Collapsed statement and associate to all the different execution plans of the SQL statements that were grouped together.

About viewing an execution plan

To view the execution plan of the statement and understand how the optimizer chooses to perform the statement, launch to the SQL tab. Each branch in the execution plan represents one operation. Each operation is displayed using the following format:

```
[<execution order>]<operation type>(<options>)<accessed object> [<cost in percentage>]
```

When examining the execution plan, always start by exploring it from right to left and top to bottom (according to the execution order in a branch on the tree). The first operation to be performed has execution order 1, followed by 2, and so on.

About estimating the cost of an execution plan

By observing the access plan of a selected statement or batch, you can analyze the access path chosen by the SQL Server Optimizer.

From the Instance Details area, in the Dashboard tab, you can observe information reported for the top statements found by the SmarTune process. You can launch directly from a finding in the Findings table to the SQL tab to compare the costs of the execution plan operators and quickly locate the most resource-consuming operation in the execution plan.

Operations are evaluated as follows:

- Statements with total estimated costs greater than the predefined thresholds (where 1 is the default) are checked for critical operations.
- Operations whose estimated cost percentage is greater than a predefined threshold (where 20% is the default) are marked as critical.

The information displayed in the Operations tab helps you understand the nature of the execution plan, its major operations, and the major access types to objects (such as, index scan, index seek, and table scan).

About analyzing changes and their effect on the statement's execution plan

Any change made to the schema or configuration parameters can adversely affect the performance of your application. Precise for SQL Server allows you track the schema and configuration changes and compare them with the resource consumption of your statement to understand how the changes related to the statement affected its performance and execution plan.

Check the Findings table in the SmarTune tab to view information on schema changes. Select a finding and launch to the SQL History view to observe the effect these changes had on resource consumption.

As you can see schema changes, volume changes, and database option changes, were made to objects referred by this statement. By examining the time the changes were reported and when the statement's performance began to deteriorate, you can locate the changes that may have been the cause of the degradation in statement performance. In the Association area you can view additional details regarding the changes and you can launch from a specific change to another tab to further your investigation. (For example: by clicking on the **Launch** icon of **Instance parameter Changes** you will launch to Object tab while displaying the **Instance Parameters Changes** in the Main Area.

About recommending indexes for an SQL statement

Precise for SQL Server uses the Microsoft® SQL Server 2000 Index Tuning Wizard (ITWIZ) and the Microsoft® SQL Server 2005 Database Tuning Adviser (DTA) to obtain recommended indexes and statistics for a selected statement or batch. The Recommended Indexes/Statistics list is displayed in the list on the left. Detailed information for each recommended item, such as a list of key columns and the DDL “create index/statistic” text, is displayed in the information tabs on the right.

If you are running SQL Server 2005, use the What-If option to show the statements whose performance would improve and the statements whose performance would deteriorate by implementing the recommendation.

About object tuning

In many cases, you may want to tune a database object rather than a specific statement that accesses the database object. This is especially true if you were not the one to write the application (for example, ERP and CRM applications) and therefore cannot change the text of any statement. In this case you will want to identify which objects are the best tuning candidates.

About identifying the best candidate for object tuning

In the Explore Objects tab in the Objects tab, you can examine the performance of active objects and determine which objects are good candidates for tuning. You can observe whether an object is an index or table, the name of the database it is associated with, a summary of the time it spent in SQL Server, and its size (in pages). If the object is an SQL Server 2005 object, additional information regarding I/O operations and waits, contentions, index usability, and logical, user and system operations is also displayed.

Clicking on the launch icon, next to an object, launches to the Recommend tab where you can view recommendations on the selected table.

You can also drill down on an object and view additional information on the selected object in the Main area of the Explore Object tab. For example, you can drill down to associated SQL statements. This helps you identify the top resource-consuming SQL statements that are accessing a selected table or index.

You can view all SQL statements that use the indexes built on the specific table. Drilling down from an index to all SQL statements using the selected index allows you to determine whether or not the specific index is actually being used. Precise for SQL Server also displays a list of unused indexes for a table or database.

Another method you can use to investigate how an index is being used is to drill down on the index and view information on its common access patterns. If you observe that an index is being updated, but shows no seek or scan activity, this may indicate that the index is not being optimally used.

The following indicators may indicate that you need to re-evaluate your index:

- User operations are low while system operations are high, indicating that the index is only being used for maintenance purposes.
- Level of I/O contentions and the amount of statement access time is high.

About recommending an index for a table

The decision to add a specific index to a table is complex. The index can improve one statement and may negatively affect others. Therefore, before adding an index, you should examine every statement accessing the table to see how the new index will impact it.

Precise for SQL Server uses the Microsoft® SQL Server 2000 Index Tuning Wizard (ITWIZ) and the Microsoft® SQL Server 2005 Database Tuning Adviser (DTA) to obtain recommended indexes and statistics for a selected table. The recommendations are based on all the statements referring to the selected table that were executed during the selected time period that have an average duration time greater than a predefined threshold (0 seconds is the default).

The Recommended Indexes or Statistics list is displayed on the left. Detailed information for each recommended item, such as a list of key columns and the DDL “create index/statistic” text, is displayed in the information tabs on the right. In addition, if you are running SQL Server 2005, Precise for SQL Server can recommend whether to drop an index, or create a partition function or schema.

If you are running SQL Server 2005, use the What-If option to show the statements whose cost would increase and the statements whose cost would decrease if the recommendation was implemented. You can select the recommendations you want to test and run the What-If function specifically on those recommendations.

The What-If function displays a pie chart that indicates the distribution of the cost of all statements that were evaluated. The Analyzed Statements table displays the explained statements that access the objects whose recommendations are being evaluated. You can continue your analysis of a particular statement on the SQL tab Plan tab, by clicking its statement ID link.

About examining schema changes over time

The Collect Schema Changes process collects changes made in an instance configuration, databases, objects, columns, indexes, and index keys. The objects sampled are User Tables, Stored Procedures, Views, Functions (all kinds), Foreign Keys, Triggers, and Extended Stored Procedures.

Schema changes take the following aspects into consideration:

- Entities dropped
- Changes made to the existing entities (collected properties are detailed in the Objects tab)
- Entities created

The changes are displayed in a list (summarized and detailed) and an overtime graph. The changes are summarized by groups such as Schema Changes, Instance Property Changes and Database Option Changes at the Instance level and Object Change in Structure and Index at the Database level. The deeper you drill down into the sub-entities the more detailed the information you will see regarding the changes. For example, you can drill down to a specific database to observe database option changes.

In addition, the In MS-SQL overtime graph is displayed with respect to changes so that you can observe how the changes that were made affected the performance of your application.

About examining object space usage over time

Another way to tune a database object is to examine its space usage. The Collect Space Utilization process collects table and index space information daily (by default).

Space usage information allows you to examine space allocated versus space used by a table or an index. This lets you easily keep track of object growth over time and decide how to allocate memory and connect objects to file groups in the specified database.

About I/O tuning

In theory, database files should be spread over different physical disks so that the I/O for one database file will not interfere with the I/O of a different database file. As a result, tuning the I/O on a large storage machine is a complex task. Moving a database file from one disk to another can speed up some parts of the application but may slow down others.

Precise for SQL Server includes storage points that can help you tune the I/O in your system. Storage points are agents that sample activity in the Storage units against which they are installed. They map datafiles to Storage devices and provide data that relates I/O wait of database objects to the Storage devices on which they are stored.

If you are running SQL Server 2005, both I/O read and I/O write information is displayed in the Objects tab.

Storage points are extensions to Precise for SQL Server. They require an additional license.

About identifying overactive storage devices

Precise for SQL Server displays information that relates I/O wait to the EMC Storage devices in which they are stored. This allows you to locate specific EMC Storage devices where performance problems are occurring.

By analyzing the information collected from your SQL Server databases, Precise for SQL Server provides an accurate picture of I/O activity and all the information needed to tune the I/O system. By identifying which EMC Storage devices are overactive, you can maintain performance levels by reorganizing storage configurations, reallocating your database objects to other file groups, or moving file groups to alternative devices.

About identifying and balancing I/O waits on the storage device level

After identifying which storage devices suffer from performance bottlenecks, you can drill down to them and analyze the I/O waits experienced during the selected time period. This can help you identify the logical files that are most probably causing the I/O bottleneck.

About identifying the most accessed database files

When one of the storage points is installed, you can compare the I/O wait on different physical disks to find out where the bottleneck is. You can then drill down from the physical disk level to the database file level to identify the database file with the most I/O waits.

About identifying and balancing I/O waits on the physical device level

Before moving a database file to another physical disk, you can examine the I/O wait on the database file to identify the time period that is experiencing the most I/O waits. You may also want to examine I/O waits on a physical disk over time to identify which physical disk is experiencing either no I/O waits, or very few I/O waits, during the same time period.

About examining file space usage over time

The Collect Space Utilization process collects database file space information daily (by default). This lets you examine file size versus the space used by each file in a database. This can help you locate the most space-consuming file, check its rate of growth, and help you determine how to best allocate memory and arrange the database files on disks.

About instance statistics tuning

The Precise for SQL Server Collector collects a wide range of performance counters from the operating system, some which belong to SQL Server and some to the operating system (OS) itself. These performance counters are sampled every minute and displayed in the Current and Statistics tabs. When loaded into the PMDB, the counters are aggregated and kept at a slice level. The counters are then aggregated to daily, weekly and monthly levels.

About observing instance statistics over time

The Statistics tab provides many performance counters grouped into several predefined overtime graphs that enable you to locate various aspects of performance problems, such as CPU, paging, I/O, and network. The Statistics tab can be used to monitor your system's current state as well as previous states. It is also possible to monitor a specific counter associated with a particular performance group by selecting the required counter.