Data Capacity Planning

Last Modified on 04 August 2021

> **Note:** See the Apply SQL Server Data Compression article for information and T-SQL scripts to apply data compression to your SQL Sentry database.
>
> We also have the SQL Sentry Scalability Pack for large environments to take advantage of columnstore compression and performance. See the Installation Recommendations article and SentryOne & Microsoft Achieve SQL Server Monitoring Performance Goals blog post for additional information.

Performance Analysis uses the SQL Sentry database to store all the performance data it collects, utilizing a high performance storage scheme. How much space is used by **Performance Analysis** is directly dependent on:

- The number of databases on the watched SQL Servers because some of the performance counters collected by **Performance Analysis** are database specific.
- The number of physical disks on the watched targets because related counters are disk specific.
- The **Minimum Duration** specified for the **Top SQL** event source. The default global setting is five seconds, meaning that any batches or stored procedures that run for longer than five seconds are collected. If this threshold is lowered, the amount of **Top SQL** data collected increases. A different **Minimum Duration** is specified for each SQL Server.
- Whether **Collect Statement Events** is set to **True** for the **Top SQL** event source. The default is **False**. If enabled, this may increase the amount of **Top SQL** data collected by a factor of two or more. This setting is also adjustable for each SQL Server.
- The **Performance Data Retention** settings. Different settings can be specified for detailed (or raw) performance data, rolled up performance data, and **Top SQL/Blocking SQL/Deadlock** data.

For detailed performance data, retention is specified in hours for each performance counter category in the **HistoryDataRetentionHours** column of the **PerformanceAnalysisCounterCategory** table. The default may be either 48 or 72 hours, depending on the category. Raw data is shown by default on the **Dashboard** and **Disk Activity** tabs when the current date range is less than or equal to 30 minutes. Over 30 minutes, rolled up data is used.

- If you have an unusually large number of databases on SQL Servers monitored by **Performance Analysis**, consider reducing the retention hours for the **SQLSERVER:DATABASES** and **SQLPERF:VIRTUAL_FILESTATS** categories. Data for these categories are stored in the **PerformanceAnalysisDataDatabaseCounter** and **PerformanceAnalysisDataDiskCounter** tables respectively.
- If you have an unusually large number of physical disks per target monitored by **Performance Analysis**, consider reducing the retention hours for the **PHYSICALDISK** category. Data for this category is stored in the **PerformanceAnalysisDataDiskCounter** table.

- Data for all other categories is stored in the **PerformanceAnalysisData** table.
- Keep the retention hours the same for categories that are stored in the same table, otherwise page splitting and fragmentation may result during the pruning process that may eventually affect performance.
- For rolled up performance data, retention is specified in hours for each rollup level in the **HistoryDataRetentionHours** column of the **PerformanceAnalysisDataRollupLevel** table. Rollup data for each break level (specified by the **LevelBreakMinutes** column) is stored in a separate table, all named **PerformanceAnalysisDataRollupXX**, where XX represents the ID of the break level. The only rollup table that may get large is the table for the two minute break level, or **PerformanceAnalysisDataRollup2**. The retention hours for this, or any other break level, can be adjusted as needed.
- Retention for raw **Top SQL**, **Blocking**, and **Deadlock** data is controlled by the **Keep Performance History** setting under **Configuration** > **Global Settings** > **Storage** in the **Navigator** pane. The default is 15 days.
- For viewing **Performance Analysis** data on the **Events** calendar, the raw **Top SQL**, **Blocking**, and **Deadlock** data is converted to the native event storage format and stored in the **EventSourceHistory** table alongside data for other event sources like SQL Agent Jobs. Retention for all event sources is controlled by the **Keep Event History** setting under **Configuration** > **Global Settings** > **Storage** in the **Navigator** pane.

Expired performance data is pruned by the SQL Sentry monitoring service every minute or so. The default settings enable you to always have detailed performance data for the last two or three days. However, if you find that you're frequently navigating to date ranges using the **Dashboard** or **Disk Activity** tabs where no data is shown, it may mean that you need to increase the retention hours for the detailed and/or rolled up performance data. Balance any changes with the resulting impact it has on database size.

When you start using **Performance Analysis**, you'll find that your SQL Sentry database grows quickly at first. After a few days this levels off once the pruning of expired data begins and starts keeping pace with the incoming new data. Get an idea of the mix of **Performance Analysis** data in your environment by inspecting sizes for the related tables using the script below. Much of the data in **EventSourceHistory** is likely related to the **Event** sources.

Additional Information: See the Enabling Higher Resolution Performance Charts in SentryOne blog post for details on how you can dramatically increase the detail data retention when using the SQL Sentry Scalability Pack.

Performance Analysis Data Script

Additional Information: See the Performance Analysis Data Usage Query.sql file on GitHub to get a list of your data consumption for the performance analysis tables in the SQL Sentry database.

	TableName	RowCount	UsedSpaceMB	ReservedSpaceMB
1	dbo.BlockChainDetail	16130	10	11
2	dbo.EventSourceHistory	8023711	36647	38502
3	dbo.MetaHistorySharePointTimerJob	0	0	0
4	dbo.MetaHistorySqlServerBlockLog	9205	1	1
5	dbo.MetaHistorySqlServerTraceLog	4022849	223	224
6	dbo.PerformanceAnalysisData	464441454	8500	8551
7	dbo.PerformanceAnalysisDataDatabaseCounter	35247983	119	120
8	dbo.PerformanceAnalysisDataDiskCounter	150766517	422	434
9	dbo.PerformanceAnalysisDataRollup11	51303304	1015	1031
10	dbo.PerformanceAnalysisDataRollup2	730101087	14980	15049
11	dbo.PerformanceAnalysisDataRollup4	278684292	5942	5966
12	dbo.PerformanceAnalysisDataRollup6	180808831	3553	3577
13	dbo.PerformanceAnalysisDataRollup8	54168193	1071	1075
14	dbo.PerformanceAnalysisPlan	7355	80	115
15	dbo.PerformanceAnalysisPlanOpTotals	387933	61	89
16	dbo.PerformanceAnalysisSsasCubeDimensionAttribute	6367	2	5
17	dbo.PerformanceAnalysisSsasTraceDataDetail	15816	3	3
18	dbo.PerformanceAnalysisSsasUsageTotals	238086	110	113
19	dbo.PerformanceAnalysisTraceCachedPlanItems	226386	75	124
20	dbo.PerformanceAnalysisTraceData	6498653	78420	79571
21	dbo.PerformanceAnalysisTraceDataToCachedPlans	981155	139	164
22	dbo.PerformanceAnalysisTraceQueryStats	58877	67	103

Query results example

Note: The results above are from a SQL Sentry database where the SQL Sentry Scalability Pack is installed.