

What Did that SQL Query Do?

SQL Server Edition

When investigating database performance issues related to SQL queries, it is often useful to know the execution plan of the statement. However, unless it was executed recently, we only have the execution plan that we can generate via a show plan command. This may or may not be the actual plan that was used when the SQL ran. Using Foglight Performance Investigator for SQL Server, we can easily go back and review *actual* plans that were used when the statement was executed. In addition, we can compare different plans as they change over time due to database object changes, optimizer statistics, etc.



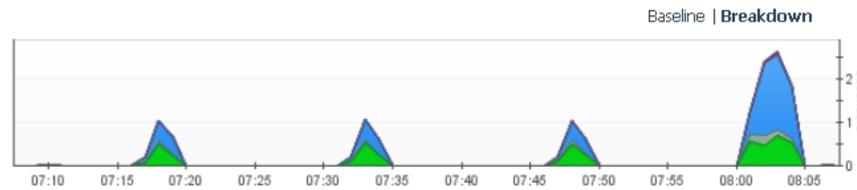
Performance Tree

- Instance View
 - SQL Statements
 - update LogShippingHistory set R...
 - SELECT MAX(SessionID)+1 FROM
 - DELETE [dbo].[QUEST_SC_MSS
 - select * from LogShippingHistory
 - if exists(select * from LogShippir
 - update LogShippingHistory set R
 - select @session_id = isnull(max(s
 - xp_restore_log
 - select target_data from sys.dm
 - Encrypted
 - insert into spotlight_playback_da
 - xp_enumerrorlogs
 - select top 100 DatabaseName, F
 - select * from OpenRowset(TrcD
 - insert into #_89D6B73B30E64A
 - insert into #temp_trace select t
 - Select top(200) T1.[name] As j
 - with wait_types (waittype, wait
 - COMMIT TRA
 - Query Hash: 0xb75fd62500000C
 - Encrypted
 - SELECT TOP 1000 SYNTAX_KEY
 - insert into LitespeedBackupFile (
 - Encrypted
 - SELECT top(5000) Convert(NVa
 - TSQL Batches

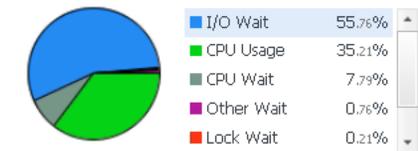
History | Change Tracking

Dimension Filter: Instance View > SQL Statements

Resource Consumption



Resource Breakdown



Top Wait Events

Overview | Blocking History

Top SQL Statements

Select Metric | View Full Text | Analyze Plan | Tune SQL | Compare | Drill to SQL Statement | Export to CSV

SQL Statements	Active Time	CPU Usage	Avg Res
update LogShippingHistory set ReplicateID = @P1 from LogShipping... (nolock) where ReplicateID is null) and ReplicateID is null	224.03	86.88	
update LogShippingHistory set ReplicateID = @P1 where ReplicateID = @P2	79.35	67.94	
SELECT MAX(SessionID)+1 FROM LogShippingHistory	157.22	50.62	
select * from LogShippingHistory with (nolock) where ReplicateID = @P1	101.64	38.61	
if exists(select * from LogShippingHistory with (nolock) where ReplicateID is null)	99.42	38.27	
select target_data from sys.dm_xe_sessions s join sys.dm_xe_sess...cdw05_3843' and t.target_name = N'ring_buffer' --stop session	2.70	2.25	
DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] FROM [dbo].[QUEST_SC...E_KEY] AND TD.[PYRAMID_LEVEL] = @P1 AND TD.[START_TIME] < @P2	138.55	1.79	
select @session_id = isnull(max(session_id),0) + 1 from msdb.dbo...id = @agent_id and agent_type = @agent_type -- -- all done --	3.82	1.05	

In the screenshot above, a time period was isolated and the top SQL statements sorted by CPU Usage are shown. To review the execution plans, all you need to do is pick a SQL and click "Analyze Plan".



1/14/16 12:55 AM | Actual | OX06000800bdbdef294021dacc0000000000000000...

Compare Plans | Generate Plan | Open in SSMS

Statement

```
(@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT].[..._KEY] AND TD.[PY] = @P1 AND TD.[START_TIME] < @P2
```

Plan Analysis

Total cost: 49.0206000 | Total I/O cost: 44.8041363 | Total CPU cost: 4.2164807

Plan Details | Operator Analysis | Object Analysis

Operator	Object	Operator Cost	Subtree Cost
Clustered Index Delete (Delete)	dbo.QUEST_SC_MSSQL_SQL_STAT_FACT.QUEST_SC_MSSQL_SQL_STAT_FACT_I99	17.40 %	49.0206000
Top		0.00 %	40.4889000
Sort		1.93 %	40.4875000
Hash Match (Inner Join)		5.38 %	39.5434000
Index Seek	dbo.QUEST_TIME_DIM.QUEST_TIME_DIM_I91	0.01 %	0.0033313
Clustered Index Scan	dbo.QUEST_SC_MSSQL_SQL_STAT_FACT.QUEST_SC_MSSQL_SQL_STAT_FACT_I99	75.28 %	36.9005000

SQL Text

```
(@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] FROM [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] SQF INNER JOIN [dbo].[QUEST_TIME_DIM] TD ON SQF.[TIME_KEY] = TD.[TIME_KEY] AND TD.[PY]
```

This drilldown shows you Plan Details, Operator Analysis and Object Analysis on separate tabs.



1/14/16 12:55 AM | Actual | 0x06000800bdbdef294021dacc000000000000...

Compare Plans | Generate Plan | Open in SSMS

Statement

((@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT..._KEY] AND TD.[PY = @P1 AND TD.[START_TIME] < @P2

Plan Analysis

Total cost: 49.0206000 | Total I/O cost: 44.8041363 | Total CPU cost: 4.2164807

Plan Details | Operator Analysis | Object Analysis

Operator	Operator Cost	Rows	I/O Cost	CPU Cost
Clustered Index Scan	75.28 %	570,517	36.2728000	0.6277260
Clustered Index Delete	17.40 %	14,681	8.5169500	0.0146813
Hash Match	5.38 %	14,681	0.0000000	2.6395500
Sort	1.93 %	14,681	0.0112613	0.9328490
Index Seek	0.01 %	45	0.0031250	0.0002062
Top	0.00 %	14,681	0.0000000	0.0014681

SQL Text

((@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] FROM [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] SQF INNER JOIN [dbo].[QUEST_TIME_DIM] TD ON SQF.[TIME_KEY] = TD.[TIME_KEY] AND TD.[PY



1/14/16 12:55 AM | Actual | 0x06000800bdbdef294021dacc0000000000000000...

Compare Plans | Generate Plan | Open in SSMS

Statement

```
(@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT..._KEY] AND TD.[PY = @P1 AND TD.[START_TIME] < @P2
```

Plan Analysis

Total cost: 49.0206000 | Total I/O cost: 44.8041363 | Total CPU cost: 4.2164807

Plan Details | Operator Analysis | Object Analysis

Name	Database	Type	Associated Operators
dbo.QUEST_SC_MSSQL_SQL_STAT_FACT.QUEST_SC_MSSQL_SQL_STAT_FACT_IX99	Quest_Performance_Repository	Index (Clustered)	Clustered Index Delete, Clustered
dbo.QUEST_TIME_DIM.QUEST_TIME_DIM_IX1	Quest_Performance_Repository	Index (NonClustered)	Index Seek

SQL Text

```
(@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] FROM [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] SQF INNER JOIN [dbo].[QUEST_TIME_DIM] TD ON SQF.[TIME_KEY] = TD.[TIME_KEY] AND TD.[PY
```

If there is more than 1 historical execution plan available, the “Compare Plans” button will be active. You can click into that drilldown to compare historical plans. Note that there will only be a historical plan stored for the first time it was detected as being changed.

Foglight™ Dell Software STC - Database Demo

Search dbadmin 1

Databases > SQL Performance > Execution Plan

Thursday, January 28, 2016 7:09:43 AM - 8:09:43 AM 60 minutes Reports

ALVSCDW05-SQL2008 Summary SQL Performance Memory Activity Databases Services HADR Logs Configuration User-defined

Powered by SQL PI

1/14/16 12:55 AM Actual 0x06000800bdbdef294021dacc00000000000000000000

Select Plan

Date	Type	Plan Handle
1/14/16 12:55 AM	Actual	0x06000800bdbdef294021dacc00000000000000000000
12/28/15 7:07 AM	Actual	0x0600080063c17e04402120370100000000000000000000
12/21/15 3:07 AM	Actual	0x0600080063c17e04402120370100000000000000000000
11/21/15 12:22 AM	Actual	0x0600080063c17e04402120370100000000000000000000
10/22/15 1:02 PM	Actual	0x0600080063c17e04402120370100000000000000000000
9/22/15 8:11 AM	Actual	0x0600080063c17e04402120370100000000000000000000
9/21/15 4:02 AM	Actual	0x0600080063c17e04402120370100000000000000000000
8/22/15 12:26 AM	Actual	0x0600080063c17e04402120370100000000000000000000
8/17/15 8:02 PM	Actual	0x0600080063c17e04402120370100000000000000000000

Analyze execution plan Cancel

Statement

```
(@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT..._KEY] AND T
=@P1 AND TD.[START_TIME] < @P2
```

Operator Cost Subtree Cost

17.40 %	49.0206000
0.00 %	40.4889000
1.93 %	40.4875000
5.38 %	39.5434000
0.01 %	0.0033313
75.28 %	36.9005000

SQL Text

```
(@P1 int,@P2 datetime)DELETE [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] FROM [dbo].[QUEST_SC_MSSQL_SQL_STAT_FACT] SQF INNER JOIN [dbo].[QUEST_TIME_DIM] TD ON SQF.[TIME_KEY] = TD.[TIME_KEY] AND TD.[PY
```

You may also analyze a specific plan by picking from the date/time dropdown shown above. This is great functionality to have because it lets you review plans that may have executed weeks or months ago, and also makes it very easy for you to detect when a plan has changed. If you also manage Oracle databases, I've posted a similar tutorial on the site as well.