Gems to help you troubleshoot query performance

Pedro Lopes
Senior Program Manager

Microsoft

SQL Server Tiger Team

Pedro Lopes
@sqlpto
pedro.lopes@microsoft.com

Senior Program Manager

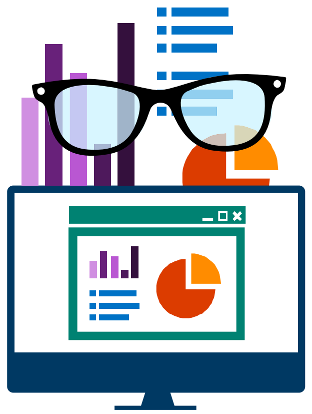
Focused on SQL Server Relational Engine

7+ years at Microsoft

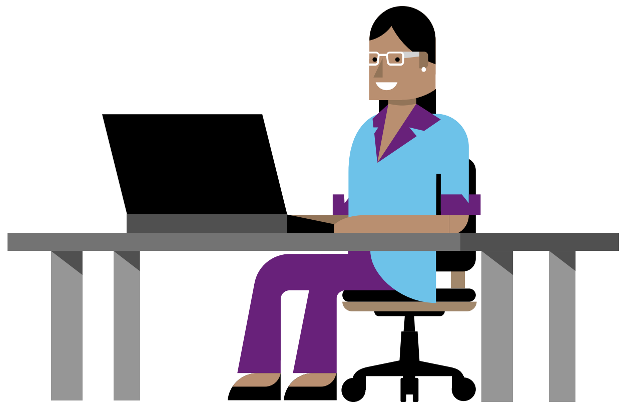
14+ years with SQL Server

Session Objectives And Takeaways

SQL Server Tiger Team



Show new diagnostics improvements for SQL Server engine



Learn how to use the new diagnostics to troubleshoot common performance issues

SQL Server Tiger Team

When was the last time you dealt with a **query** performance issue?

SQL Server Tiger Team

Query Performance Troubleshooting Fundamentals

Why does a query slow down?

- Excessive resource consumption
- Poor indexing strategy
- Lack of useful statistics
- Lack of useful partitioning
- Consequence of blocked queries
- Incorrect server configurations

Context for Slow-Running Query Analysis

- Is the performance problem related to a component other than queries?
  - For example, is the problem slow network performance?
  - Are there any other components that might be causing or contributing to performance degradation?
- If the performance issue is related to queries, which query or set of queries is involved?
- Was the query optimized with useful statistics?
- Are suitable indexes available?
- Are there any data or index hot spots?
- If you have a large volume of data, do you need to partition it?
- Is the Query Optimizer provided with the best opportunity to optimize a complex query?

SQL Server Tiger Team

How do I analyze the performance of a slow-running query?

Some tools we will use today

- SHOWPLAN XML (a.k.a. Actual Execution Plan)
- Query Store
- Plan Comparison Tool
- Live Query Stats
- xEvents
- … and more ☺

A map to the execution context

- Query plans include:
  - How data is accessed
  - How data is joined
  - Sequence of operations
  - Use of temporary worktables and sorts
  - Estimated rowcounts, iterations, and costs from each step
  - Actual rowcounts and iterations
  - How data is aggregated
  - Use of parallelism
  - Query execution warnings
  - Query execution stats
  - Hardware/Resource stats

Query and Query Plan Fingerprints

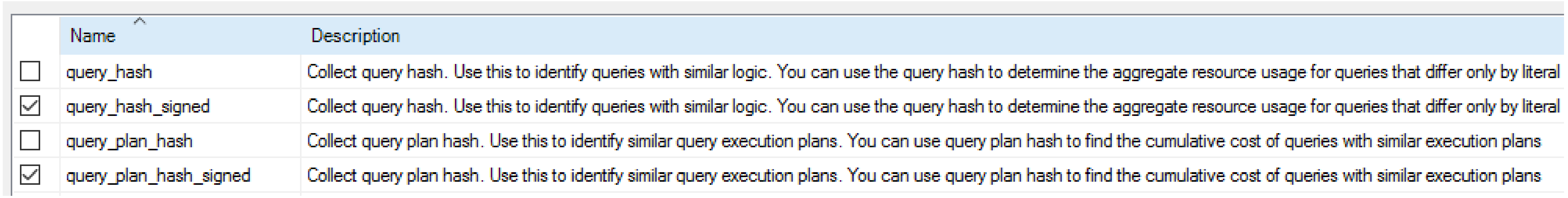
- Query Fingerprint
  - query_hash
  - Explicitly identifies a specific query in the cache.
  - *sys.dm_exec_requests*
  - *sys.dm_exec_query_stats*
- SQL Handle
  - sql_handle
  - Token for the SQL text that relates to a batch.
  - *sys.dm_exec_sql_text*
  - *sys.dm_exec_query_stats*
  - *sys.dm_exec_query_memory_grants*
- Query Plan Fingerprints
  - query_plan_hash
  - Useful to determine queries that share the same execution plan.
  - Can be used to determine if the query plan has changed.
  - *sys.dm_exec_requests*
  - *sys.dm_exec_query_stats*
- Plan Handle
  - plan_handle
  - Token for a cached execution plan.
  - *sys.dm_exec_query_plan*
  - *sys.dm_exec_cached_plans*

SQL Server Tiger Team

Diagnostic and Troubleshooting Enhancements

The pain of joining DMVs and xEvents

- *query_hash* and *query_plan_hash* actions in xEvents
  - Not the same data types as respective columns in DMVs *sys.dm_exec_requests* and *sys.dm_exec_query_stats*
  - That makes it difficult to correlate the information
- In SQL Server 2016 RTM and 2014 SP2
  - New actions *query_hash_signed* and *query_plan_hash_signed* allow you to join these DMVs with xEvents such as *rpc_completed* and *sql_batch_completed*



	Name	Description
☐	query_hash	Collect query hash. Use this to identify queries with similar logic. You can use the query hash to determine the aggregate resource usage for queries that differ only by literal
☑	query_hash_signed	Collect query hash. Use this to identify queries with similar logic. You can use the query hash to determine the aggregate resource usage for queries that differ only by literal
☐	query_plan_hash	Collect query plan hash. Use this to identify similar query execution plans. You can use query plan hash to find the cumulative cost of queries with similar execution plans
☑	query_plan_hash_signed	Collect query plan hash. Use this to identify similar query execution plans. You can use query plan hash to find the cumulative cost of queries with similar execution plans

SQL Server Tiger Team

It’s all about avoiding roundtrips to collect additional information.

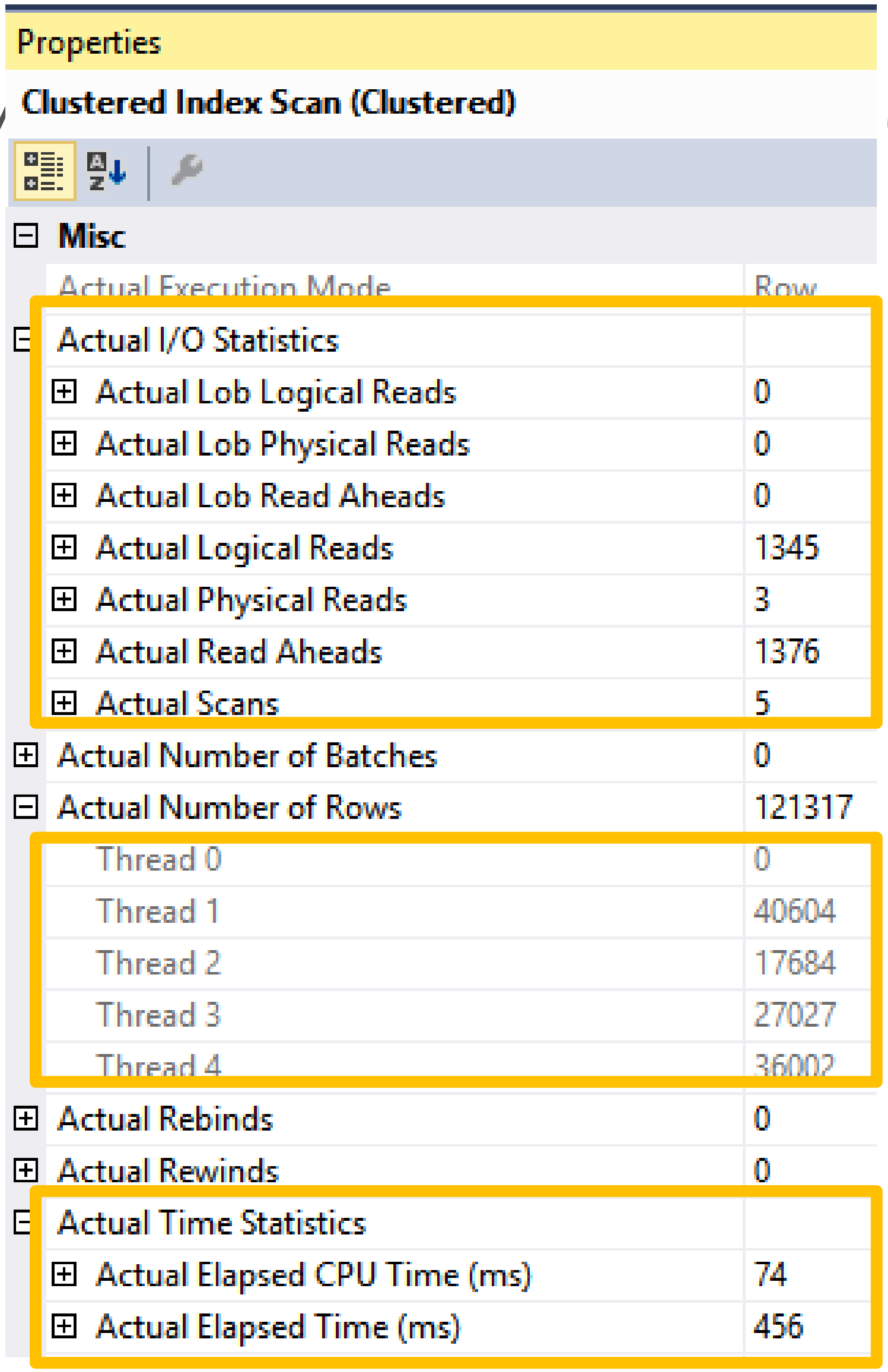
Missing perf insights on query plan nodes

- Per operator performance statistics for node and threads
- Showplan extended to include *RunTimeCountersPerThread*
- Node costs for parent and children:
  - Cumulative values for Row mode operators
  - Singleton values for Batch mode operators

Runtime Info	Up to SQL 2016	SQL 2016 / SQL 2014 SP2
`ActualRows`	X	X
`ActualRowsRead`		X
`Batches`		X
`ActualEndOfScans`	X	X
`ActualExecutions`	X	X
`ActualExecutionMode`		X
`ActualElapsedms`		X
`ActualCPUms`		X
`ActualScans`		X
`ActualLogicalReads`		X
`ActualPhysicalReads`		X
`ActualReadAheads`		X
`ActualLobLogicalReads`		X
`ActualLobPhysicalReads`		X
`ActualLobReadAheads`		X
`InputMemoryGrant`		
`OutputMemoryGrant`		
`UsedMemoryGrant`		

M on query plan nodes

SQL Server Tiger Team



Properties

Clustered Index Scan (Clustered)

Misc	
Actual Execution Mode	Row
Actual I/O Statistics	
Actual Lob Logical Reads	0
Actual Lob Physical Reads	0
Actual Lob Read Aheads	0
Actual Logical Reads	1345
Actual Physical Reads	3
Actual Read Aheads	1376
Actual Scans	5
Actual Number of Batches	0
Actual Number of Rows	121317
Thread 0	0
Thread 1	40604
Thread 2	17684
Thread 3	27027
Thread 4	36002
Actual Rebinds	0
Actual Rewinds	0
Actual Time Statistics	
Actual Elapsed CPU Time (ms)	74
Actual Elapsed Time (ms)	456

```
<RunTimeInformation>
<RunTimeCountersPerThread Thread="0" ActualRows="121317"
ActualRowsRead="10000000" Batches="0" ActualEndOfScans="3"
ActualExecutions="1" ActualExecutionMode="Row"
ActualElapsedms="456" ActualCPUms="74" ActualScans="3"
ActualLogicalReads="1345" ActualPhysicalReads="3"
ActualReadAheads="1376" ActualLobLogicalReads="0"
ctualLobPhysicalReads="0" ActualLobReadAheads="0" />
<    TimeInformation>
```

SET STATISTICS IO not needed

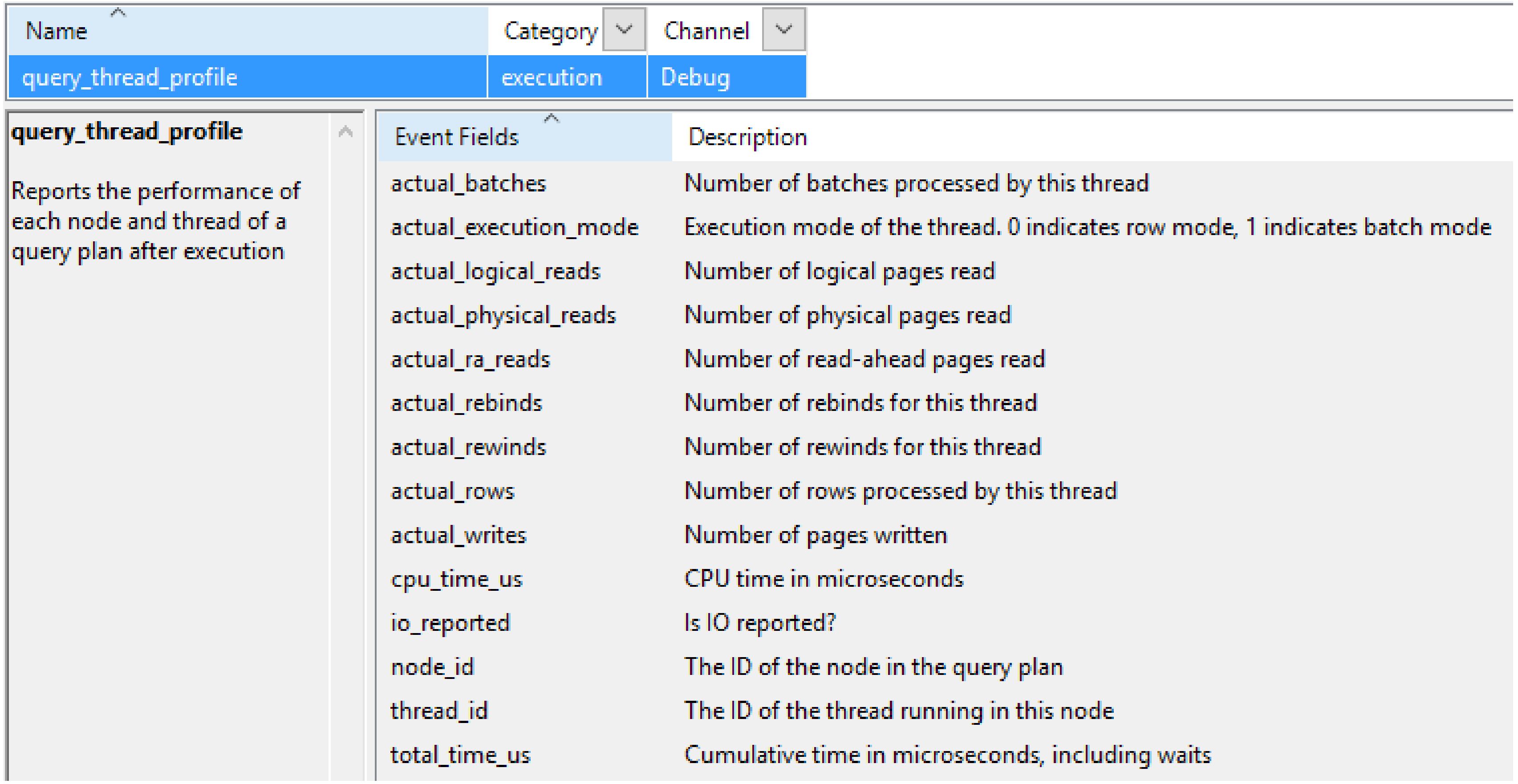
SET STATISTICS TIME not needed

Per-operator level performance stats xEvent

New Extended Event query_thread_profile

Showplan time scale = milliseconds

xEvent time scale = microseconds for CPU and total time.



Name	Category	Channel
query_thread_profile	execution	Debug

query_thread_profile

Reports the performance of each node and thread of a query plan after execution

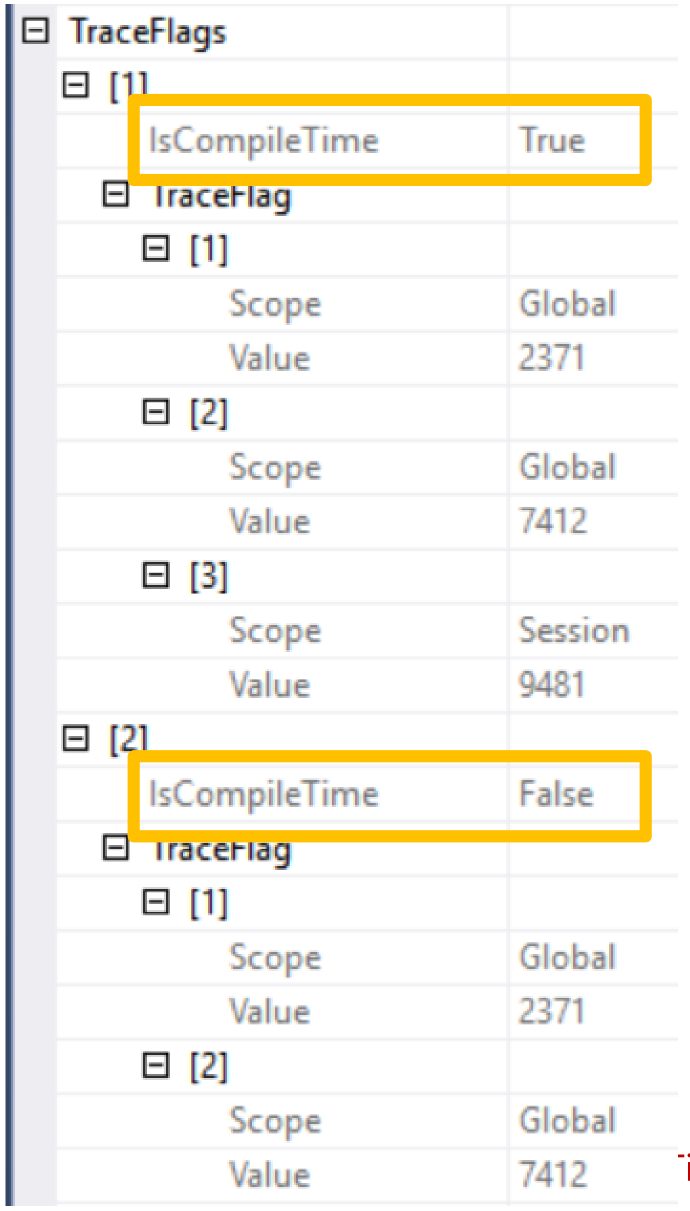
Event Fields	Description
actual_batches	Number of batches processed by this thread
actual_execution_mode	Execution mode of the thread. 0 indicates row mode, 1 indicates batch mode
actual_logical_reads	Number of logical pages read
actual_physical_reads	Number of physical pages read
actual_ra_reads	Number of read-ahead pages read
actual_rebinds	Number of rebinds for this thread
actual_rewinds	Number of rewinds for this thread
actual_rows	Number of rows processed by this thread
actual_writes	Number of pages written
cpu_time_us	CPU time in microseconds
io_reported	Is IO reported?
node_id	The ID of the node in the query plan
thread_id	The ID of the thread running in this node
total_time_us	Cumulative time in microseconds, including waits

Getting all context info in Showplan: Trace Flags

Shows list of active trace flags:

- Query
- Session
- Global

Useful to understand if active Trace Flags influence execution context



⊟ TraceFlags	
⊟ [1]	
IsCompileTime	True
⊟ TraceFlag	
⊟ [1]	
Scope	Global
Value	2371
⊟ [2]	
Scope	Global
Value	7412
⊟ [3]	
Scope	Session
Value	9481
⊟ [2]	
IsCompileTime	False
⊟ TraceFlag	
⊟ [1]	
Scope	Global
Value	2371
⊟ [2]	
Scope	Global
Value	7412

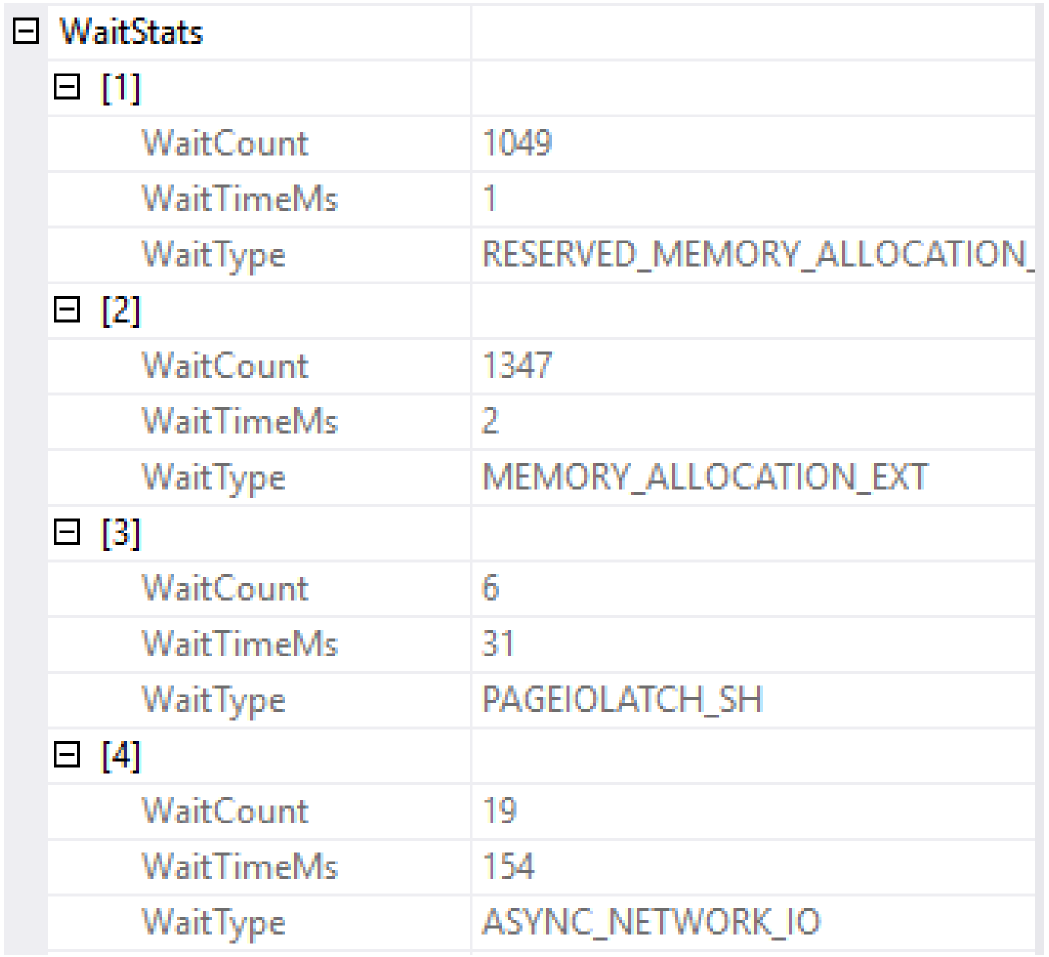
SQL Server Tiger Team

All-up Trace Flag list

http://aka.ms/traceflags

Getting all context info in Showplan: Waits

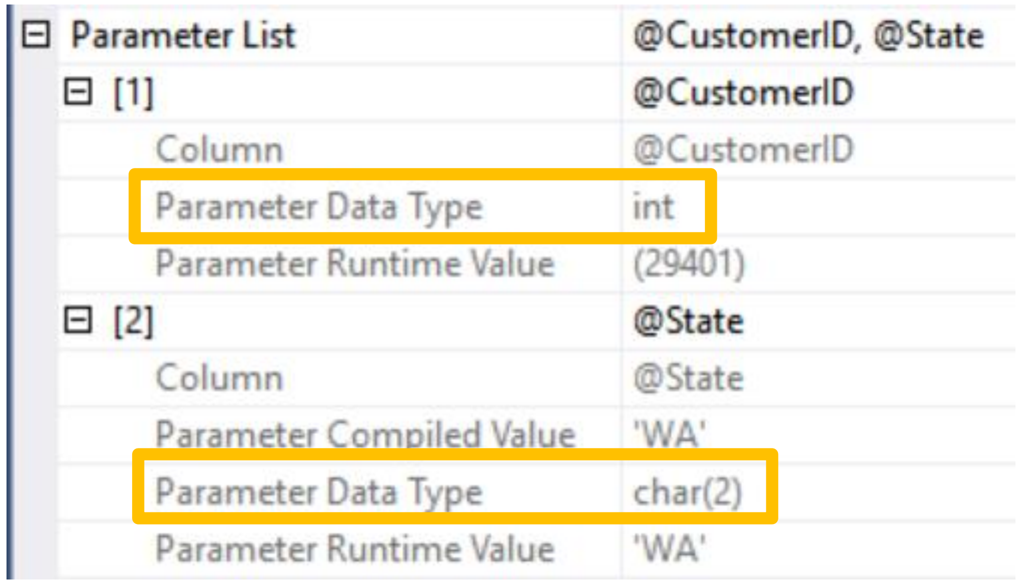
Shows top 10 waits from sys.dm_exec_session_wait_stats



⊟ WaitStats	
⊟ [1]	
WaitCount	1049
WaitTimeMs	1
WaitType	RESERVED_MEMORY_ALLOCATION_
⊟ [2]	
WaitCount	1347
WaitTimeMs	2
WaitType	MEMORY_ALLOCATION_EXT
⊟ [3]	
WaitCount	6
WaitTimeMs	31
WaitType	PAGEIOLATCH_SH
⊟ [4]	
WaitCount	19
WaitTimeMs	154
WaitType	ASYNC_NETWORK_IO

Getting all context info in Showplan: Parameter Data Types

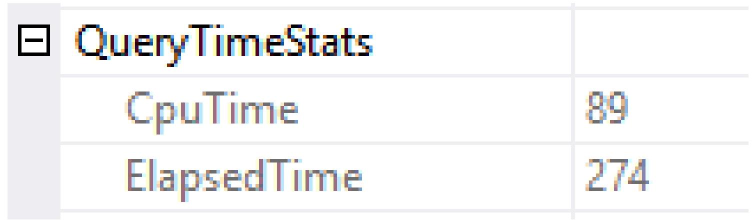
Easier detection of type conversion issues



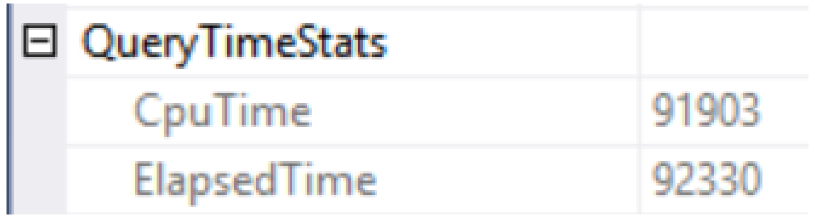
⊟ Parameter List	@CustomerID, @State
⊟ [1]	@CustomerID
Column	@CustomerID
Parameter Data Type	int
Parameter Runtime Value	(29401)
⊟ [2]	@State
Column	@State
Parameter Compiled Value	'WA'
Parameter Data Type	char(2)
Parameter Runtime Value	'WA'

Getting all context info in Showplan: Times

Persisting information on elapsed and CPU times



QueryTimeStats	
CpuTime	89
ElapsedTime	274

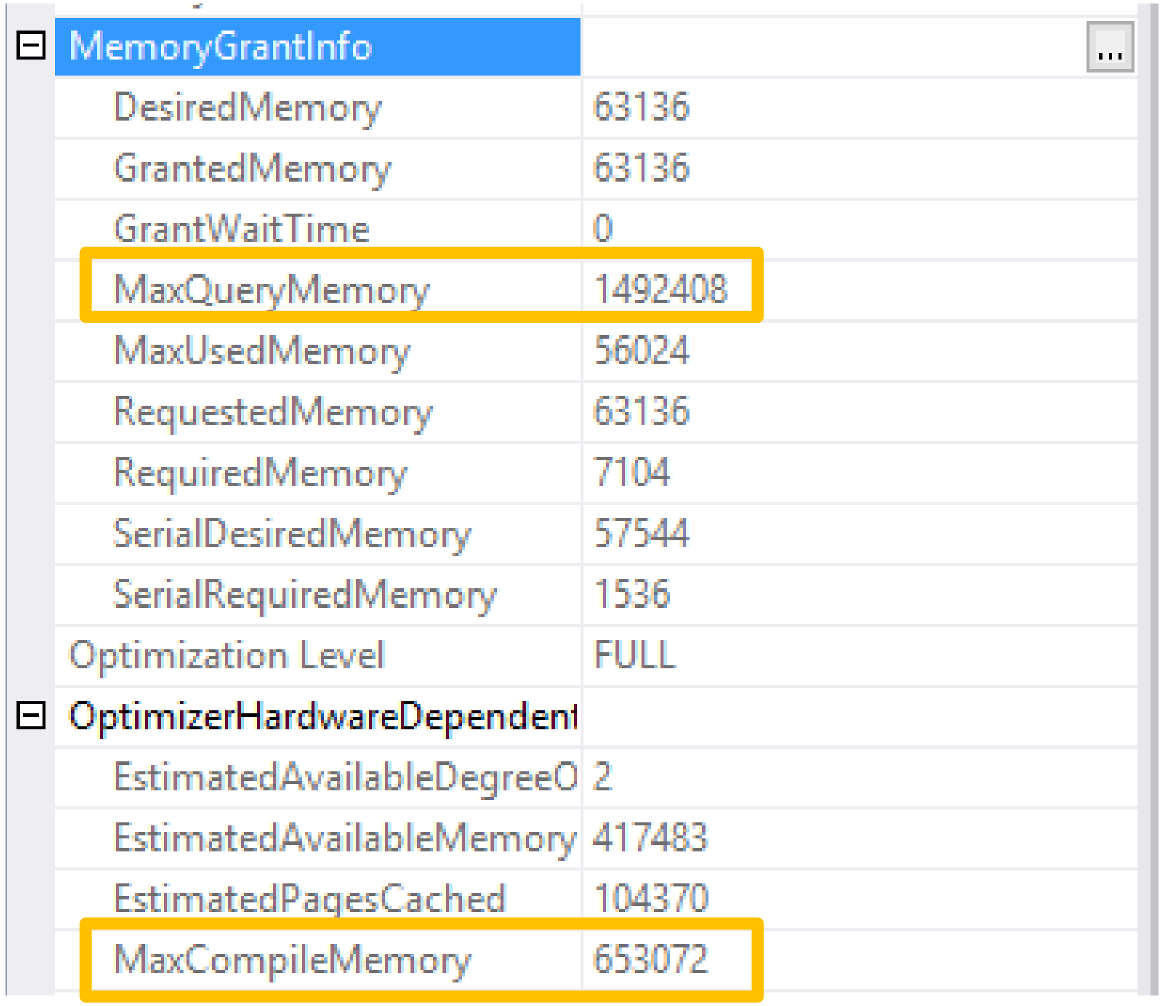


QueryTimeStats	
CpuTime	91903
ElapsedTime	92330

Getting all context info in Showplan: RG info

List attributes of Resource Governor Settings

- MaxCompileMemory for maximum query optimizer memory in KB during compile under RG
- MaxQueryMemory for maximum query memory grant under RG MAX_MEMORY_PERCENT hint



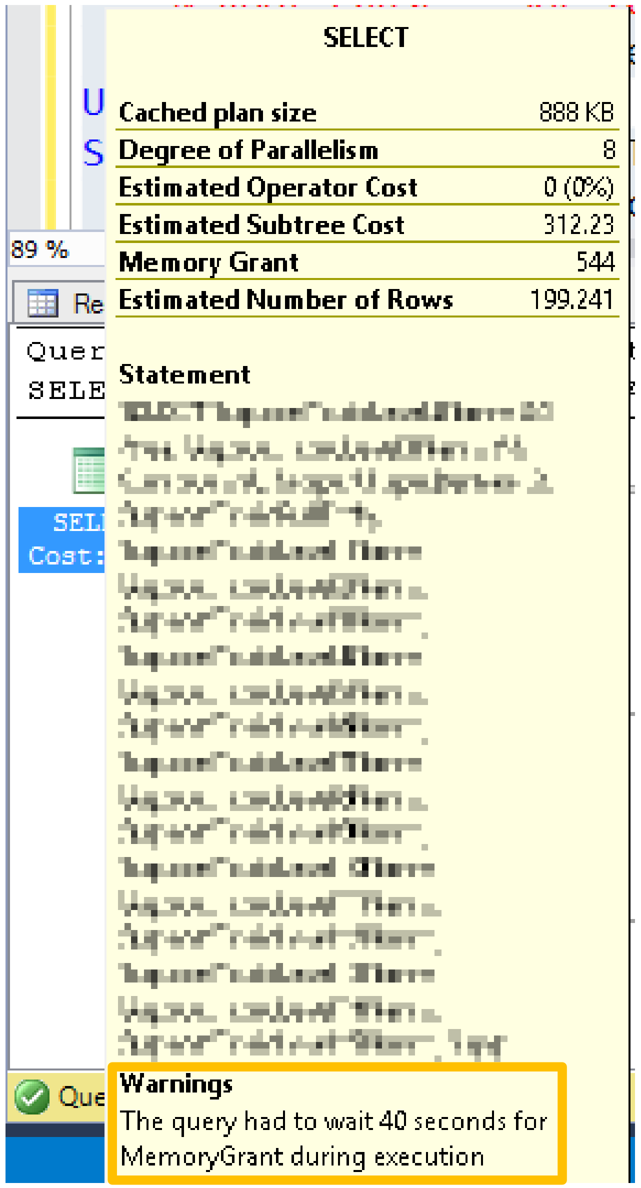
MemoryGrantInfo	
DesiredMemory	63136
GrantedMemory	63136
GrantWaitTime	0
MaxQueryMemory	1492408
MaxUsedMemory	56024
RequestedMemory	63136
RequiredMemory	7104
SerialDesiredMemory	57544
SerialRequiredMemory	1536
Optimization Level	FULL
OptimizerHardwareDependent	
EstimatedAvailableDegreeO	2
EstimatedAvailableMemory	417483
EstimatedPagesCached	104370
MaxCompileMemory	653072

SQL Server Tiger Team

Demo

Per-operator level performance stats

Memory Grant Wait Showplan Warning



SELECT

Cached plan size	888 KB
Degree of Parallelism	8
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	312.23
Memory Grant	544
Estimated Number of Rows	199.241

Statement

[illegible]

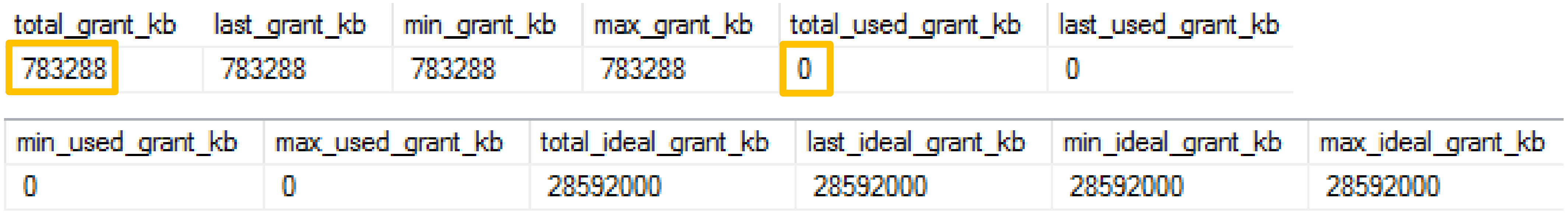
Warnings

The query had to wait 40 seconds for MemoryGrant during execution

- Occurs when a T-SQL statement or stored procedure waits more than one second for a memory grant or when the initial attempt to get memory fails.
- Since SQL Server 2012

Understanding memory usage per execution

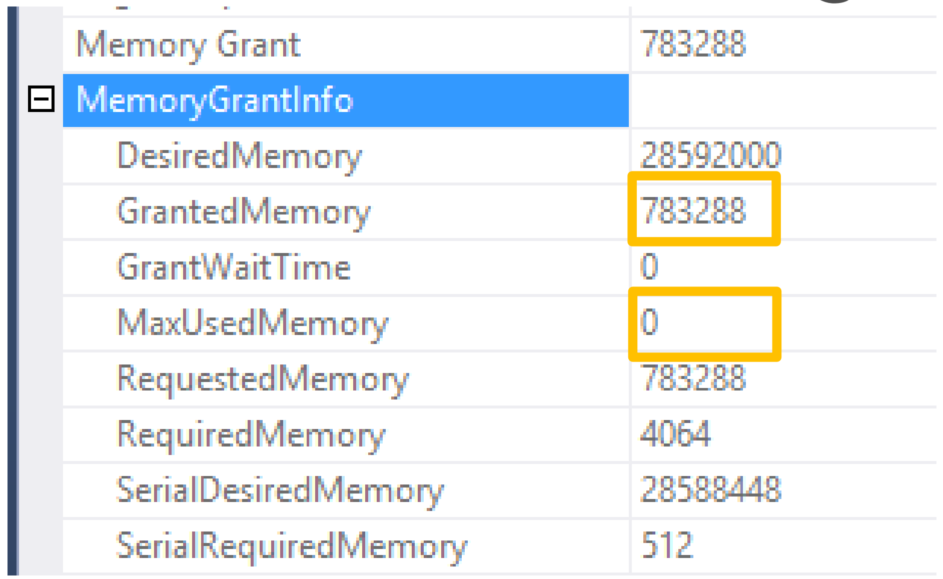
New columns in *sys.dm_exec_query_stats*



total_grant_kb	last_grant_kb	min_grant_kb	max_grant_kb	total_used_grant_kb	last_used_grant_kb
783288	783288	783288	783288	0	0

min_used_grant_kb	max_used_grant_kb	total_ideal_grant_kb	last_ideal_grant_kb	min_ideal_grant_kb	max_ideal_grant_kb
0	0	28592000	28592000	28592000	28592000

Showplan extended to include grant usage per thread and iterator



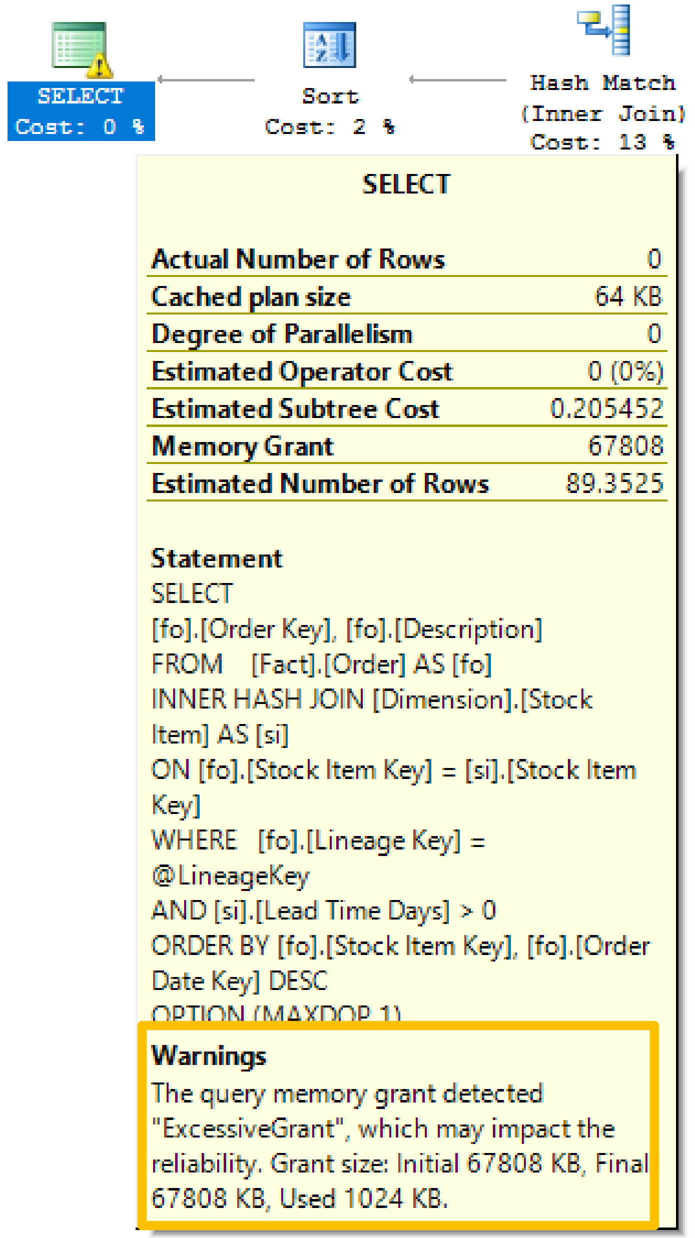
Memory Grant	783288
MemoryGrantInfo	
DesiredMemory	28592000
GrantedMemory	783288
GrantWaitTime	0
MaxUsedMemory	0
RequestedMemory	783288
RequiredMemory	4064
SerialDesiredMemory	28588448
SerialRequiredMemory	512

New Memory Grant Showplan Warning

SQL Server 2014 SP2 and SQL Server 2016 SP1

- 3 conditions:
  - **Excessive Grant**: when max used memory is too small compared to the granted memory. This scenario can cause blocking and less efficient usage when large grants exist and a fraction of that memory was used.

KB 3172997



SELECT

Actual Number of Rows	0
Cached plan size	64 KB
Degree of Parallelism	0
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.205452
Memory Grant	67808
Estimated Number of Rows	89.3525

Statement

```
SELECT
[fo].[Order Key], [fo].[Description]
FROM    [Fact].[Order] AS [fo]
INNER HASH JOIN [Dimension].[Stock
Item] AS [si]
ON [fo].[Stock Item Key] = [si].[Stock Item
Key]
WHERE   [fo].[Lineage Key] =
@LineageKey
AND [si].[Lead Time Days] > 0
ORDER BY [fo].[Stock Item Key], [fo].[Order
Date Key] DESC
OPTION (MAXDOP 1)
```

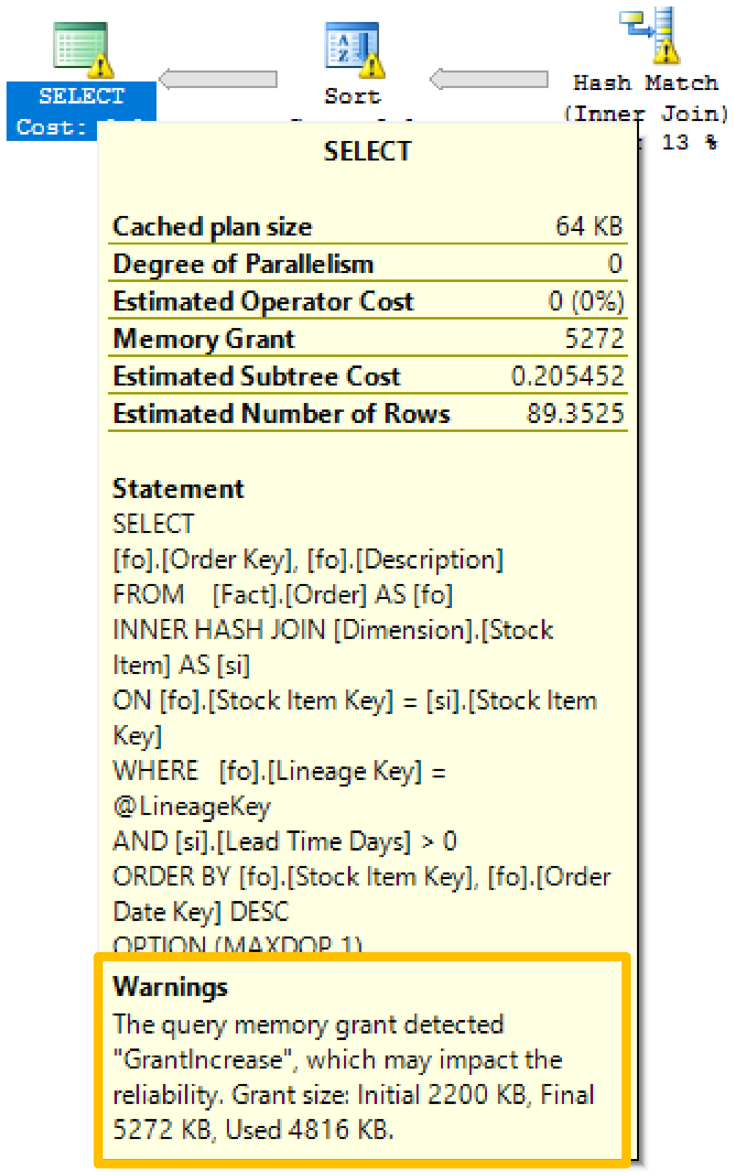
Warnings

The query memory grant detected "ExcessiveGrant", which may impact the reliability. Grant size: Initial 67808 KB, Final 67808 KB, Used 1024 KB.

New Memory Grant Showplan Warning

SQL Server 2014 SP2 and SQL Server 2016 SP1

- 3 conditions:
  - **Excessive Grant**: when max used memory is too small compared to the granted memory. This scenario can cause blocking and less efficient usage when large grants exist and a fraction of that memory was used.
  - **Grant Increase**: when the dynamic grant starts to increase too much, based on the ratio between the max used memory and initial request memory. This scenario can cause server instability and unpredictable workload performance.
  - **Used More Than Granted**: when the max used memory exceeds the granted memory. This scenario can cause OOM conditions on the server.



SELECT

Cached plan size	64 KB
Degree of Parallelism	0
Estimated Operator Cost	0 (0%)
Memory Grant	5272
Estimated Subtree Cost	0.205452
Estimated Number of Rows	89.3525

Statement

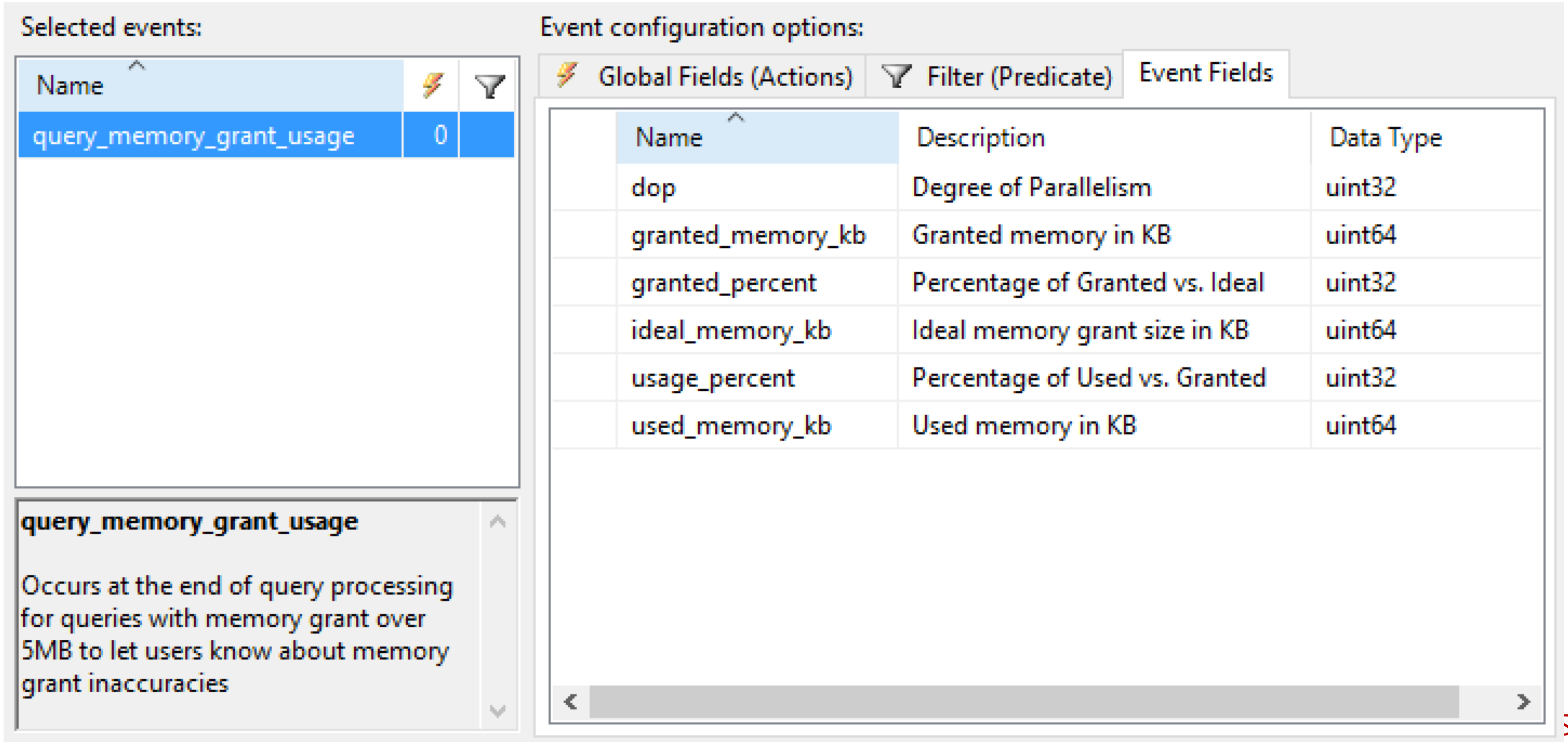
```
SELECT
[fo].[Order Key], [fo].[Description]
FROM    [Fact].[Order] AS [fo]
INNER HASH JOIN [Dimension].[Stock
Item] AS [si]
ON [fo].[Stock Item Key] = [si].[Stock Item
Key]
WHERE   [fo].[Lineage Key] =
@LineageKey
AND [si].[Lead Time Days] > 0
ORDER BY [fo].[Stock Item Key], [fo].[Order
Date Key] DESC
OPTION (MAXDOP 1)
```

Warnings

The query memory grant detected "GrantIncrease", which may impact the reliability. Grant size: Initial 2200 KB, Final 5272 KB, Used 4816 KB.

New Memory Grant xEvent

- Query_memory_grant_usage XE in SQL Server 2016
  - Detect inaccurate or insufficient memory grant, when grant is >5MB as minimum



Selected events:

Name		
query_memory_grant_usage	0	

query_memory_grant_usage

Occurs at the end of query processing for queries with memory grant over 5MB to let users know about memory grant inaccuracies

Event configuration options:

Name	Description	Data Type
dop	Degree of Parallelism	uint32
granted_memory_kb	Granted memory in KB	uint64
granted_percent	Percentage of Granted vs. Ideal	uint32
ideal_memory_kb	Ideal memory grant size in KB	uint64
usage_percent	Percentage of Used vs. Granted	uint32
used_memory_kb	Used memory in KB	uint64

min and max query memory grant option

SQL Server 2016 and SQL Server 2014 SP2

- User control over min and max grant size in percentages
  - OPTION (MAX_GRANT_PERCENT=0.1), meaning 0.1% of max allowed query memory under Resource Governor
  - The valid value is between 0 and 100%
  - MAX_GRANT_PERCENT >= MIN_GRANT_PERCENT
- Why use a floating point value?
  - On a high end machine with 1 TB of memory, 1% can be already 10GB

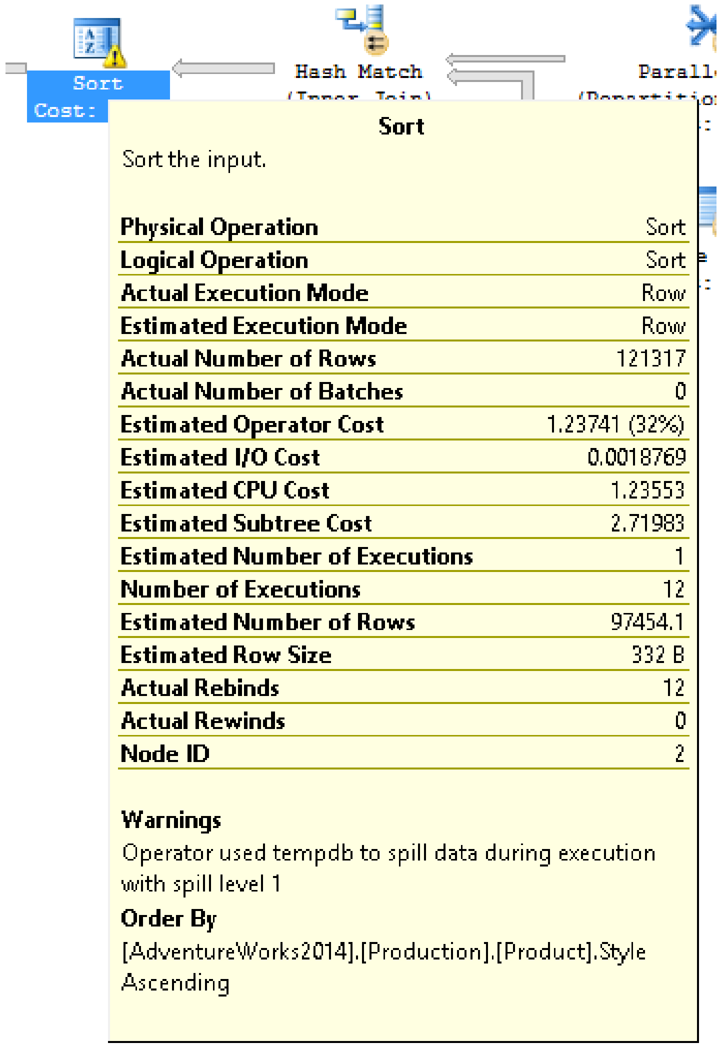
SQL Server Tiger Team

Demo

Memory grant issue detection

New Spills Warnings - Sort

Up to SQL Server 2016



Sort

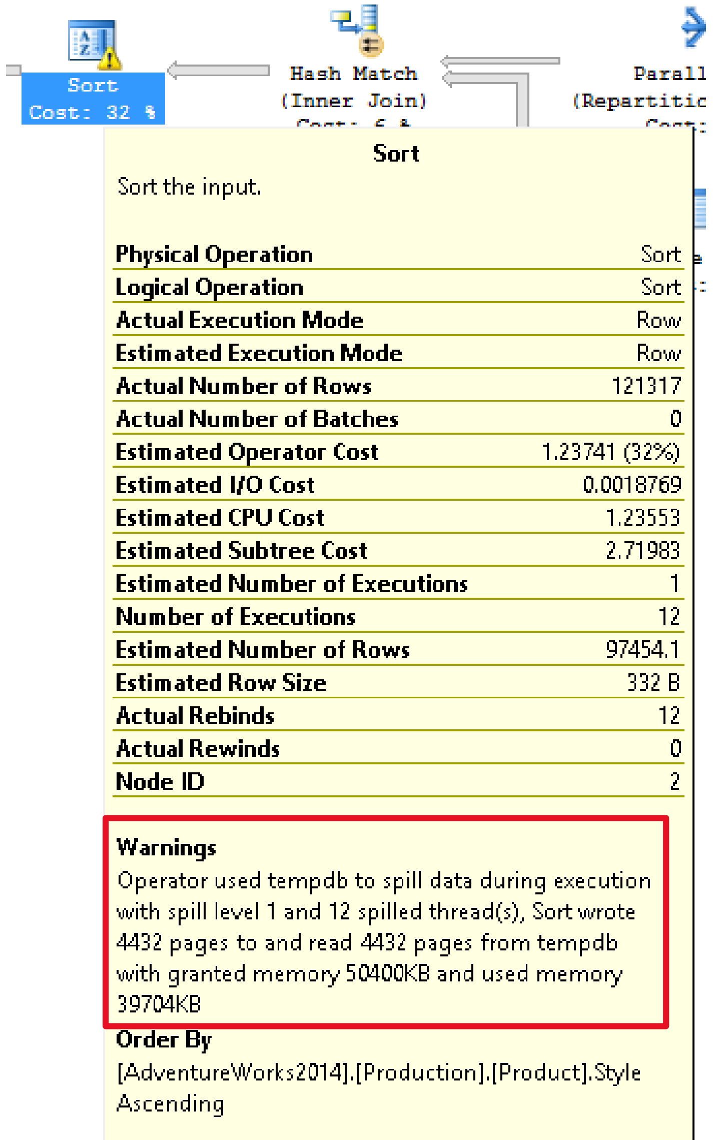
Sort the input.

Physical Operation	Sort
Logical Operation	Sort
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	121317
Actual Number of Batches	0
Estimated Operator Cost	1.23741 (32%)
Estimated I/O Cost	0.0018769
Estimated CPU Cost	1.23553
Estimated Subtree Cost	2.71983
Estimated Number of Executions	1
Number of Executions	12
Estimated Number of Rows	97454.1
Estimated Row Size	332 B
Actual Rebinds	12
Actual Rewinds	0
Node ID	2

Warnings
Operator used tempdb to spill data during execution with spill level 1

Order By
[AdventureWorks2014].[Production].[Product].Style Ascending

SQL Server 2016 and 2014 SP2



Sort

Sort the input.

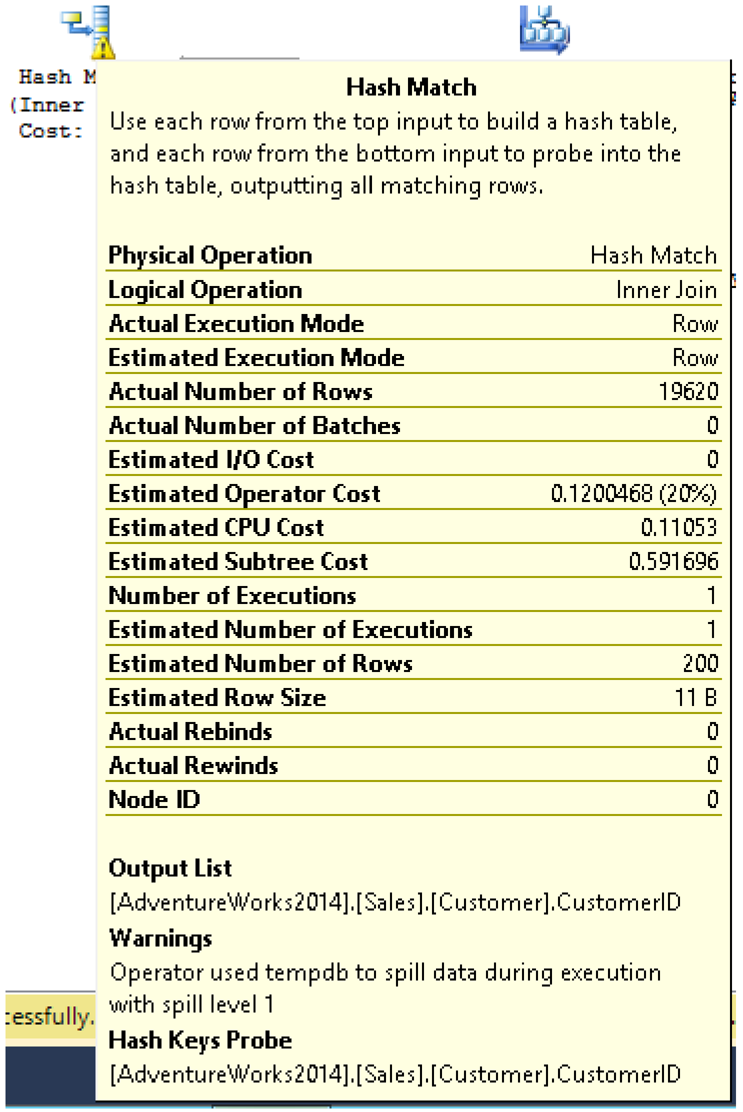
Physical Operation	Sort
Logical Operation	Sort
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	121317
Actual Number of Batches	0
Estimated Operator Cost	1.23741 (32%)
Estimated I/O Cost	0.0018769
Estimated CPU Cost	1.23553
Estimated Subtree Cost	2.71983
Estimated Number of Executions	1
Number of Executions	12
Estimated Number of Rows	97454.1
Estimated Row Size	332 B
Actual Rebinds	12
Actual Rewinds	0
Node ID	2

Warnings
Operator used tempdb to spill data during execution with spill level 1 and 12 spilled thread(s), Sort wrote 4432 pages to and read 4432 pages from tempdb with granted memory 50400KB and used memory 39704KB

Order By
[AdventureWorks2014].[Production].[Product].Style Ascending

New Spills Warnings - Hash

Up to SQL Server 2016



Hash Match
(Inner
Cost:

Hash Match

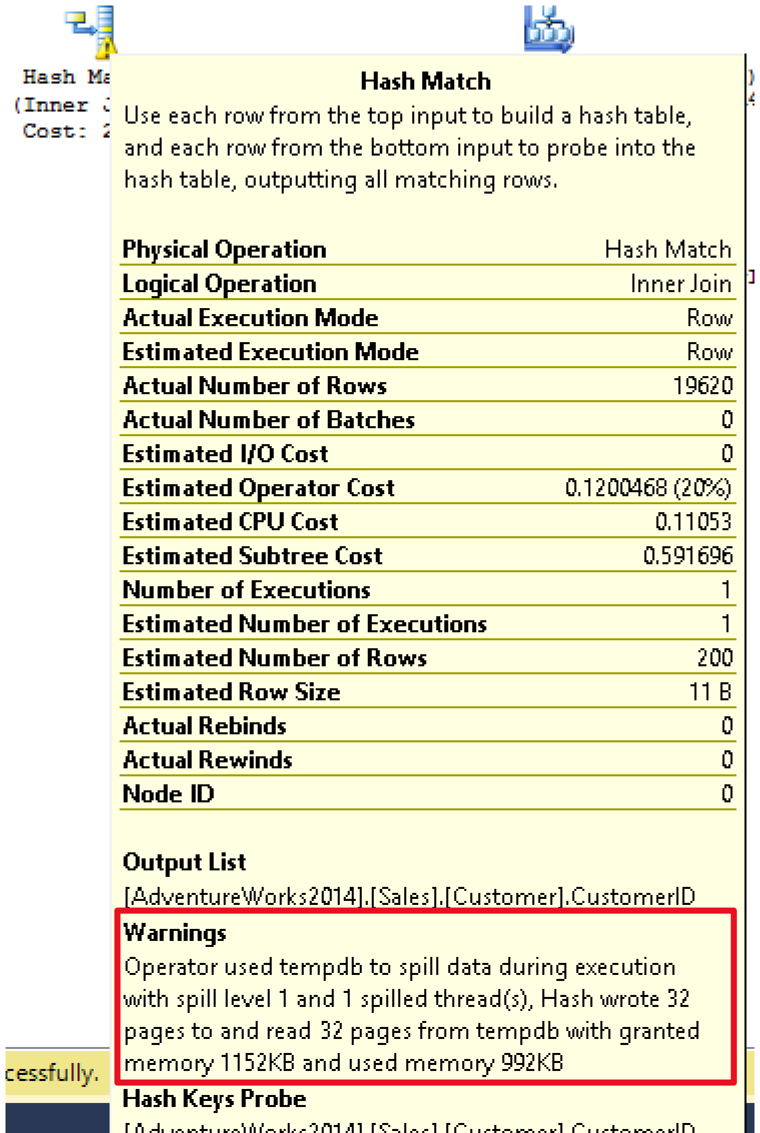
Use each row from the top input to build a hash table, and each row from the bottom input to probe into the hash table, outputting all matching rows.

Physical Operation	Hash Match
Logical Operation	Inner Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	19620
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0.1200468 (20%)
Estimated CPU Cost	0.11053
Estimated Subtree Cost	0.591696
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	200
Estimated Row Size	11 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0

Output List
[AdventureWorks2014].[Sales].[Customer].CustomerID
Warnings
Operator used tempdb to spill data during execution with spill level 1
Hash Keys Probe
[AdventureWorks2014].[Sales].[Customer].CustomerID

cessfully.

SQL Server 2016 and 2014 SP2



Hash Ma
(Inner
Cost: 2

Hash Match

Use each row from the top input to build a hash table, and each row from the bottom input to probe into the hash table, outputting all matching rows.

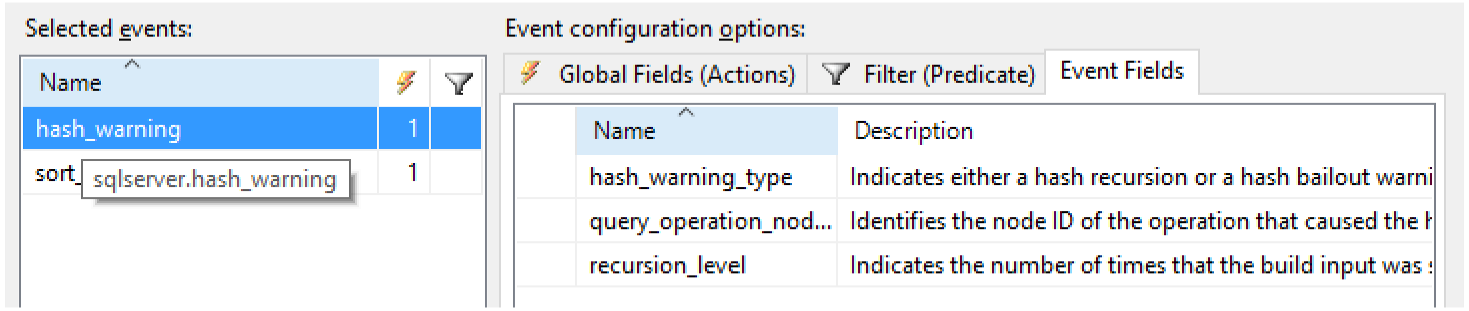
Physical Operation	Hash Match
Logical Operation	Inner Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	19620
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0.1200468 (20%)
Estimated CPU Cost	0.11053
Estimated Subtree Cost	0.591696
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	200
Estimated Row Size	11 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0

Output List
[AdventureWorks2014].[Sales].[Customer].CustomerID
Warnings
Operator used tempdb to spill data during execution with spill level 1 and 1 spilled thread(s), Hash wrote 32 pages to and read 32 pages from tempdb with granted memory 1152KB and used memory 992KB
Hash Keys Probe

cessfully.

Spill xEvents - Hash Warning

Up to SQL Server 2016



Selected events:

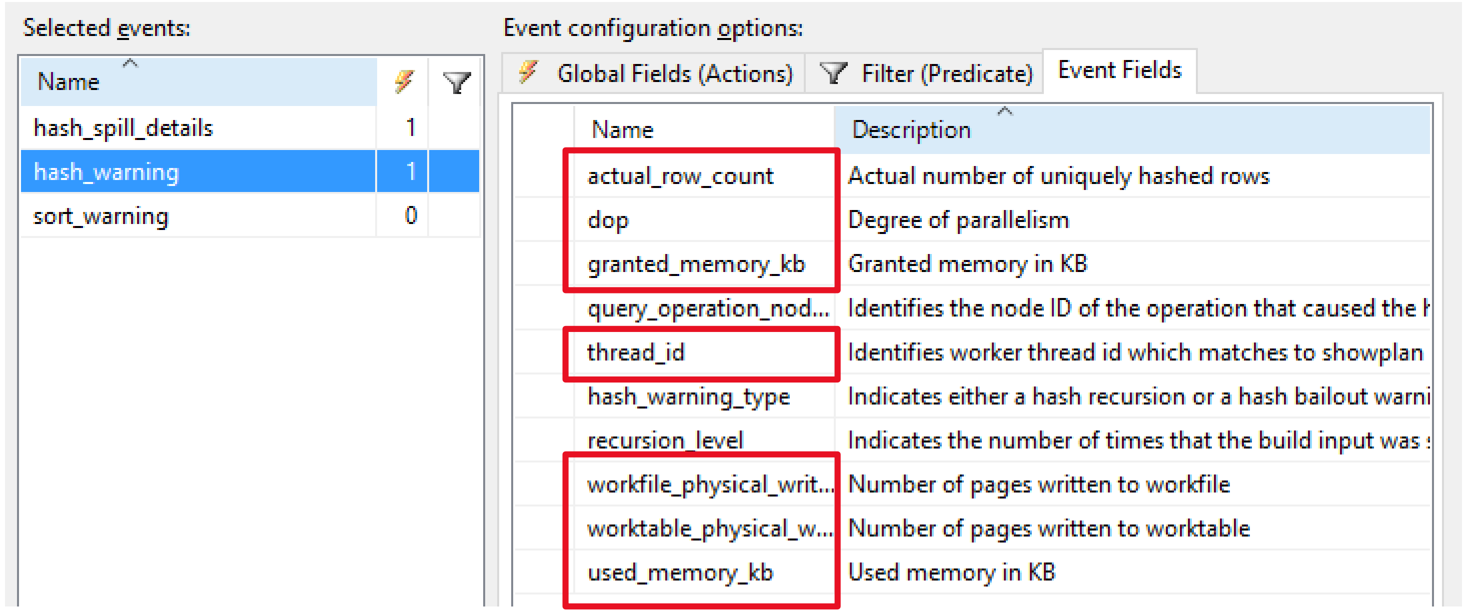
Name		
hash_warning	1	
sort_	1	

sqlserver.hash_warning

Event configuration options:

Name	Description
hash_warning_type	Indicates either a hash recursion or a hash bailout warni
query_operation_nod...	Identifies the node ID of the operation that caused the h
recursion_level	Indicates the number of times that the build input was s

SQL Server 2016 SQL Server 2014 SP2



Selected events:

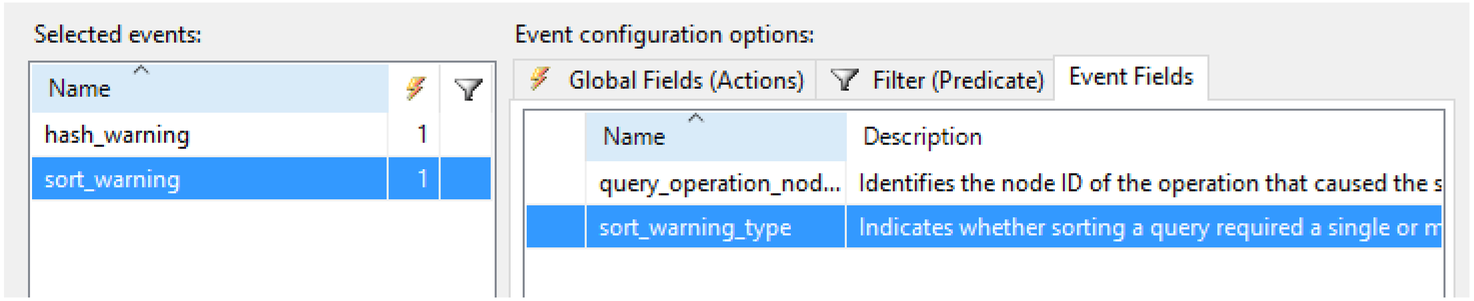
Name		
hash_spill_details	1	
hash_warning	1	
sort_warning	0	

Event configuration options:

Name	Description
actual_row_count	Actual number of uniquely hashed rows
dop	Degree of parallelism
granted_memory_kb	Granted memory in KB
query_operation_nod...	Identifies the node ID of the operation that caused the h
thread_id	Identifies worker thread id which matches to showplan
hash_warning_type	Indicates either a hash recursion or a hash bailout warni
recursion_level	Indicates the number of times that the build input was s
workfile_physical_writ...	Number of pages written to workfile
worktable_physical_w...	Number of pages written to worktable
used_memory_kb	Used memory in KB

Spill xEvents - Sort Warning

Up to SQL Server 2016



Selected events:

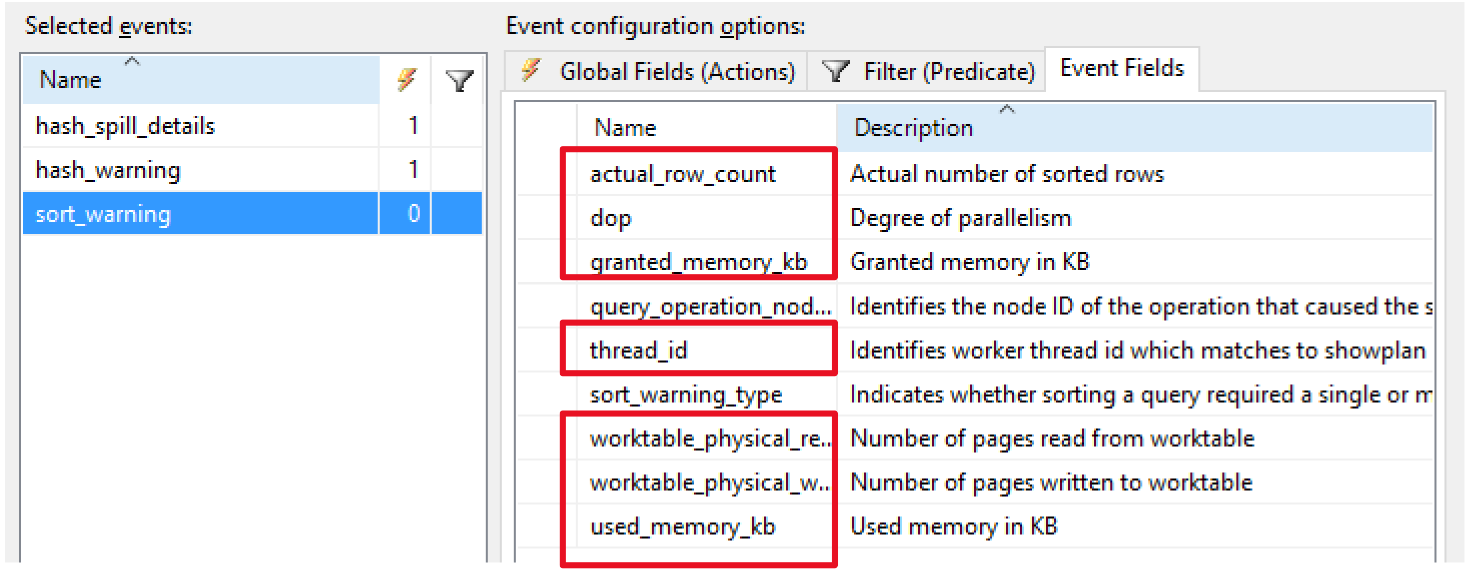
Name	⚡	▽
hash_warning	1	
sort_warning	1	

Event configuration options:

Global Fields (Actions) | Filter (Predicate) | Event Fields

Name	Description
query_operation_nod...	Identifies the node ID of the operation that caused the s
sort_warning_type	Indicates whether sorting a query required a single or m

SQL Server 2016
SQL Server 2014 SP2



Selected events:

Name	⚡	▽
hash_spill_details	1	
hash_warning	1	
sort_warning	0	

Event configuration options:

Global Fields (Actions) | Filter (Predicate) | Event Fields

Name	Description
actual_row_count	Actual number of sorted rows
dop	Degree of parallelism
granted_memory_kb	Granted memory in KB
query_operation_nod...	Identifies the node ID of the operation that caused the s
thread_id	Identifies worker thread id which matches to showplan
sort_warning_type	Indicates whether sorting a query required a single or m
worktable_physical_re...	Number of pages read from worktable
worktable_physical_w...	Number of pages written to worktable
used_memory_kb	Used memory in KB

Detecting predicate search inefficiencies?

Actual number of rows returned are rows <u>after</u> the predicate is applied.

- Not the actual number of rows that are scanned from a table or index.

Scenario hidden from an actual execution plan:

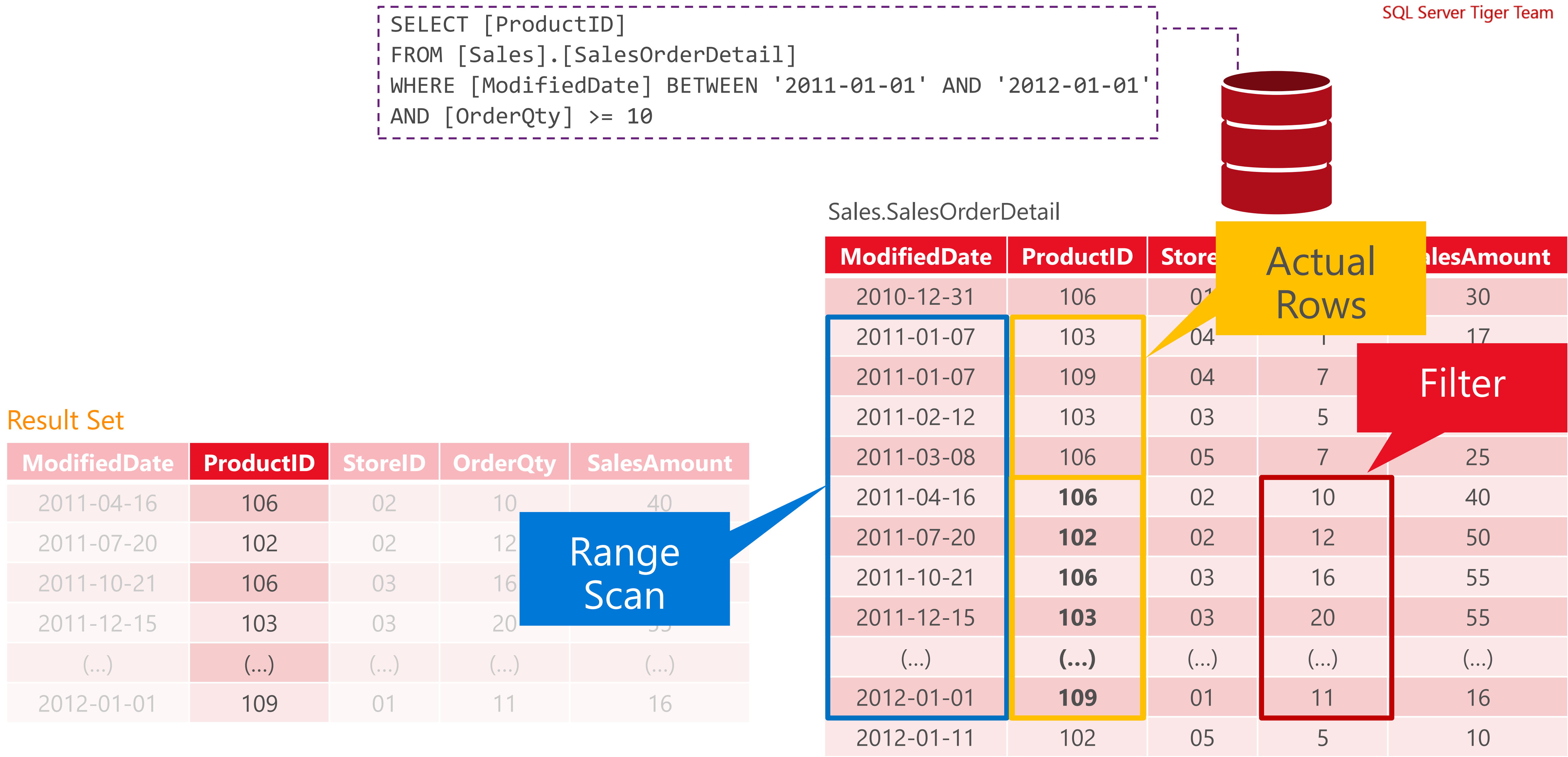
- SCAN or SEEK returns only 10 rows, why is it taking so long?
- You see high CPU or many logical reads, but the query plan doesn't reflect that.

Now what?? ☹

Searching without pushdown

SQL Server Tiger Team

```
SELECT [ProductID]
FROM [Sales].[SalesOrderDetail]
WHERE [ModifiedDate] BETWEEN '2011-01-01' AND '2012-01-01'
AND [OrderQty] >= 10
```



Sales.SalesOrderDetail

ModifiedDate	ProductID	Store		lesAmount
2010-12-31	106	01		30
2011-01-07	103	04	1	17
2011-01-07	109	04	7	
2011-02-12	103	03	5	
2011-03-08	106	05	7	25
2011-04-16	**106**	02	10	40
2011-07-20	**102**	02	12	50
2011-10-21	**106**	03	16	55
2011-12-15	**103**	03	20	55
(...)	**(...)**	(...)	(...)	(...)
2012-01-01	**109**	01	11	16
2012-01-11	102	05	5	10

Result Set

ModifiedDate	ProductID	StoreID	OrderQty	SalesAmount
2011-04-16	106	02	10	40
2011-07-20	102	02	12	
2011-10-21	106	03	16	
2011-12-15	103	03	20	55
(...)	(...)	(...)	(...)	(...)
2012-01-01	109	01	11	16

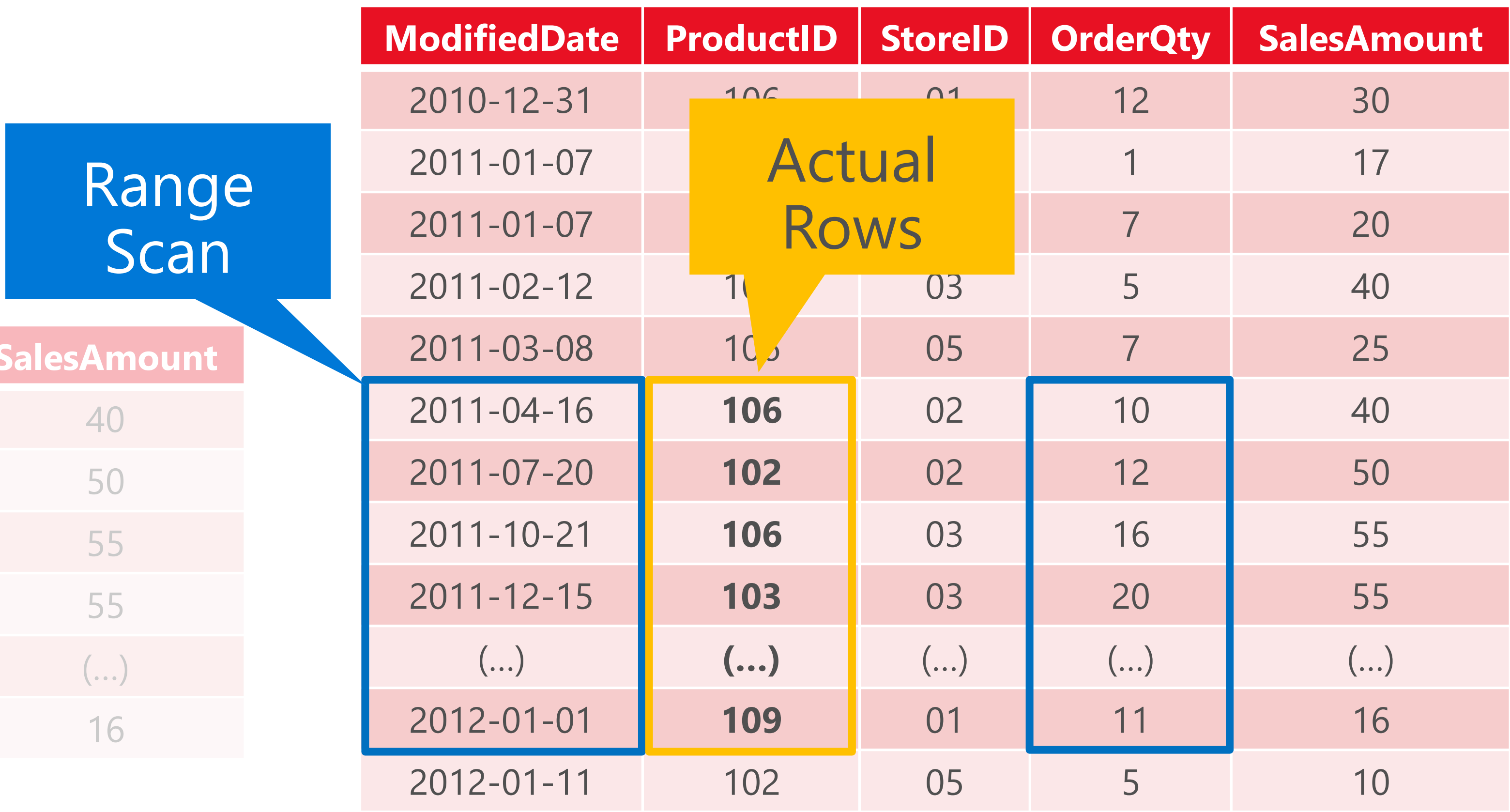
Searching with pushdown

SQL Server Tiger Team

```
SELECT [ProductID]
FROM [Sales].[SalesOrderDetail]
WHERE [ModifiedDate] BETWEEN '2011-01-01' AND '2012-01-01'
AND [OrderQty] >= 10
```

Result Set

ModifiedDate	ProductID	StoreID	OrderQty	SalesAmount
2011-04-16	106	02	10	40
2011-07-20	102	02	12	50
2011-10-21	106	03	16	55
2011-12-15	103	03	20	55
(...)	(...)	(...)	(...)	(...)
2012-01-01	109	01	11	16



Sales.SalesOrderDetail

ModifiedDate	ProductID	StoreID	OrderQty	SalesAmount
2010-12-31	106	01	12	30
2011-01-07			1	17
2011-01-07			7	20
2011-02-12	1	03	5	40
2011-03-08	10	05	7	25
2011-04-16	**106**	02	10	40
2011-07-20	**102**	02	12	50
2011-10-21	**106**	03	16	55
2011-12-15	**103**	03	20	55
(...)	**(...)**	(...)	(...)	(...)
2012-01-01	**109**	01	11	16
2012-01-11	102	05	5	10

SQL Server Tiger Team

Demo

Predicate Pushdown in Showplan

Production Alert: Application is slow!

SQL Server Tiger Team

Run data collection tools:

PSSDiag

xEvents

Profiler

…

All after the fact

Take the data:

Repro

Analyze

Deploy mitigation

Production Alert: Query Perf Issues!

What if I could do live query troubleshooting?

- To have in-flight query execution statistics, the *query execution statistics profile infrastructure* must be enabled **on demand**.
- But cost overhead goes up to 75%.
  - It makes bad things worse if running all the time.
  - This is why customers still revert back to previous pattern.

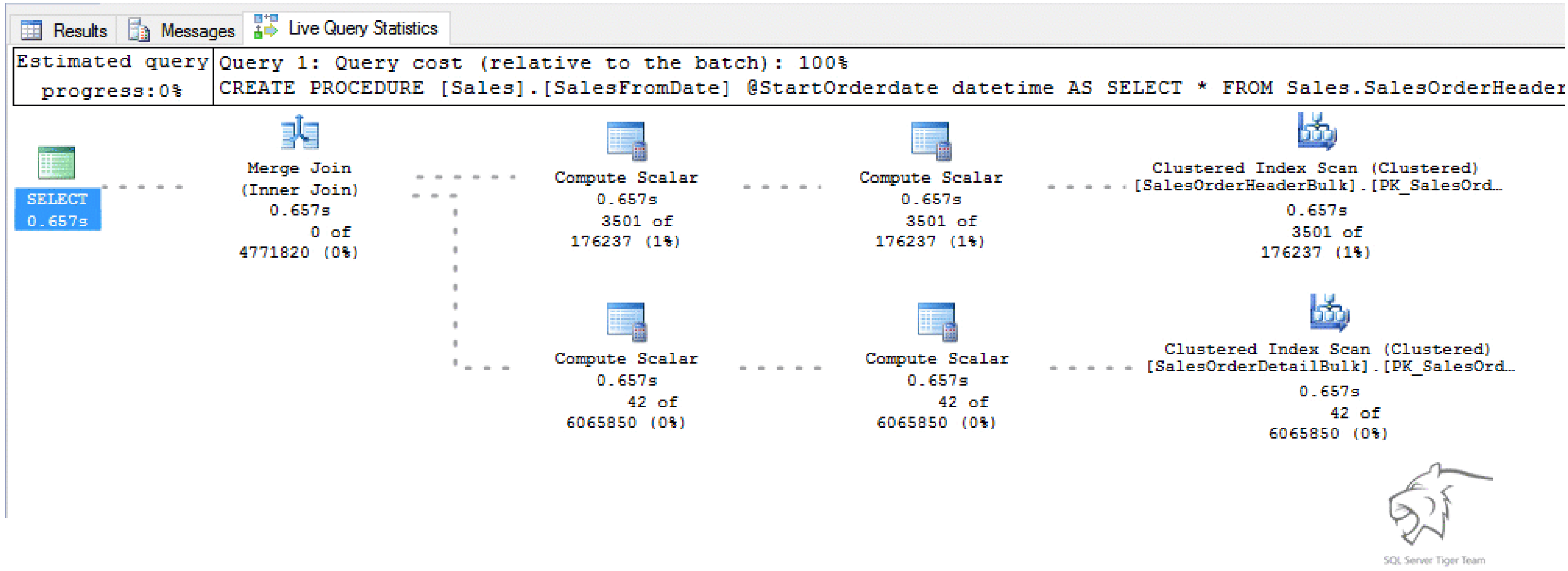
Unleash Lightweight Profiling

SQL Server Tiger Team

Tap to in-flight execution

Find the hotspot

Deploy mitigation



```
Estimated query progress:0%
Query 1: Query cost (relative to the batch): 100%
CREATE PROCEDURE [Sales].[SalesFromDate] @StartOrderdate datetime AS SELECT * FROM Sales.SalesOrderHeader

SELECT 0.657s

Merge Join (Inner Join)
0.657s
0 of
4771820 (0%)

Compute Scalar
0.657s
3501 of
176237 (1%)

Compute Scalar
0.657s
3501 of
176237 (1%)

Clustered Index Scan (Clustered)
[SalesOrderHeaderBulk].[PK_SalesOrd…
0.657s
3501 of
176237 (1%)

Compute Scalar
0.657s
42 of
6065850 (0%)

Compute Scalar
0.657s
42 of
6065850 (0%)

Clustered Index Scan (Clustered)
[SalesOrderDetailBulk].[PK_SalesOrd…
0.657s
42 of
6065850 (0%)
```

Tracking query progress (estimated)

- To have in-flight query execution statistics, the *query execution statistics profile infrastructure* must be enabled on demand.
- Can be enabled for a target session:
  - Specifying Include Live Query Statistics in SSMS.
  - SET STATISTICS XML ON
  - SET STATISTICS PROFILE ON
- Or globally to view the LQS from other sessions (such as from Activity Monitor):
  - Enabling *query_post_execution_showplan* extended event.
- High overhead (75% with TPC-C like workload)

KB
3170113

Lightweight Tracking query progress (estimated)

- Lightweight query execution profiling **dramatically reduces performance overhead** of continuously collecting per-operator query execution statistics (such as actual number of rows)
- Can be enabled by:
  - Using global TF 7412.
  - Enabling *query_thread_profile* extended event.
  - When lightweight profiling is on, *sys.dm_exec_query_profiles* is also populated for all sessions.
- This enables usage of LQS feature in SSMS (including Activity Monitor) and of the new DMF **sys.dm_exec_query_statistics_xml**.
- The following still use regular profiling infra:
  - SET STATISTICS XML (or Include Actual Plan).
  - *query_post_execution_showplan* extended event.

KB 3190871

What is the impact of live query troubleshooting?

Query Execution Statistics Profiling Infrastructure tests with TPC-C like workloads

	Overhead percent (up to)	
Infra Type	**no active xEvents**	**Active xEvent query_post_execution_showplan**
Legacy	**75.5**	93.17
Lightweight in SQL Server 2014 SP2/2016	3.5	62.02
Lightweight in SQL Server 2016 SP1	**2**	14.3

SQL Server Tiger Team

Demo

Live Query Troubleshooting

What Diagnostic choices do you have?

SQL Server Tiger Team

Regular Profiling

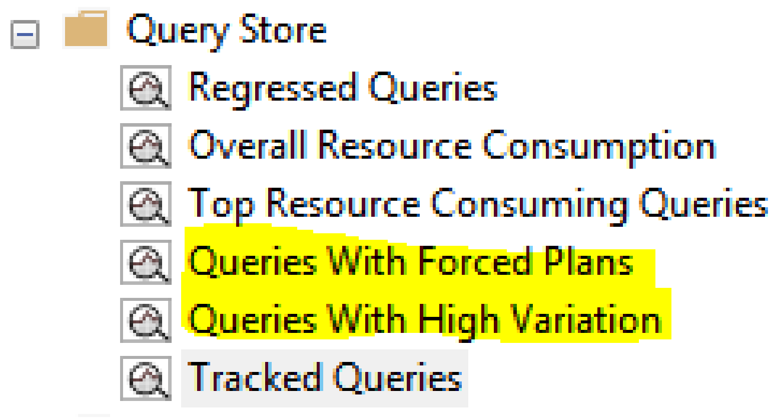
- Full runtime statistics for a query plan
- Most expensive overhead
- Can be enabled per session or globally
- Consume data from live queries or post execution

Lightweight Profiling

- Limited runtime statistics in query plan (no CPU tracking)
- Least expensive overhead
- Only enabled globally
- Consume data from live queries or post execution
- More optimized in SQL Server 2016 SP1

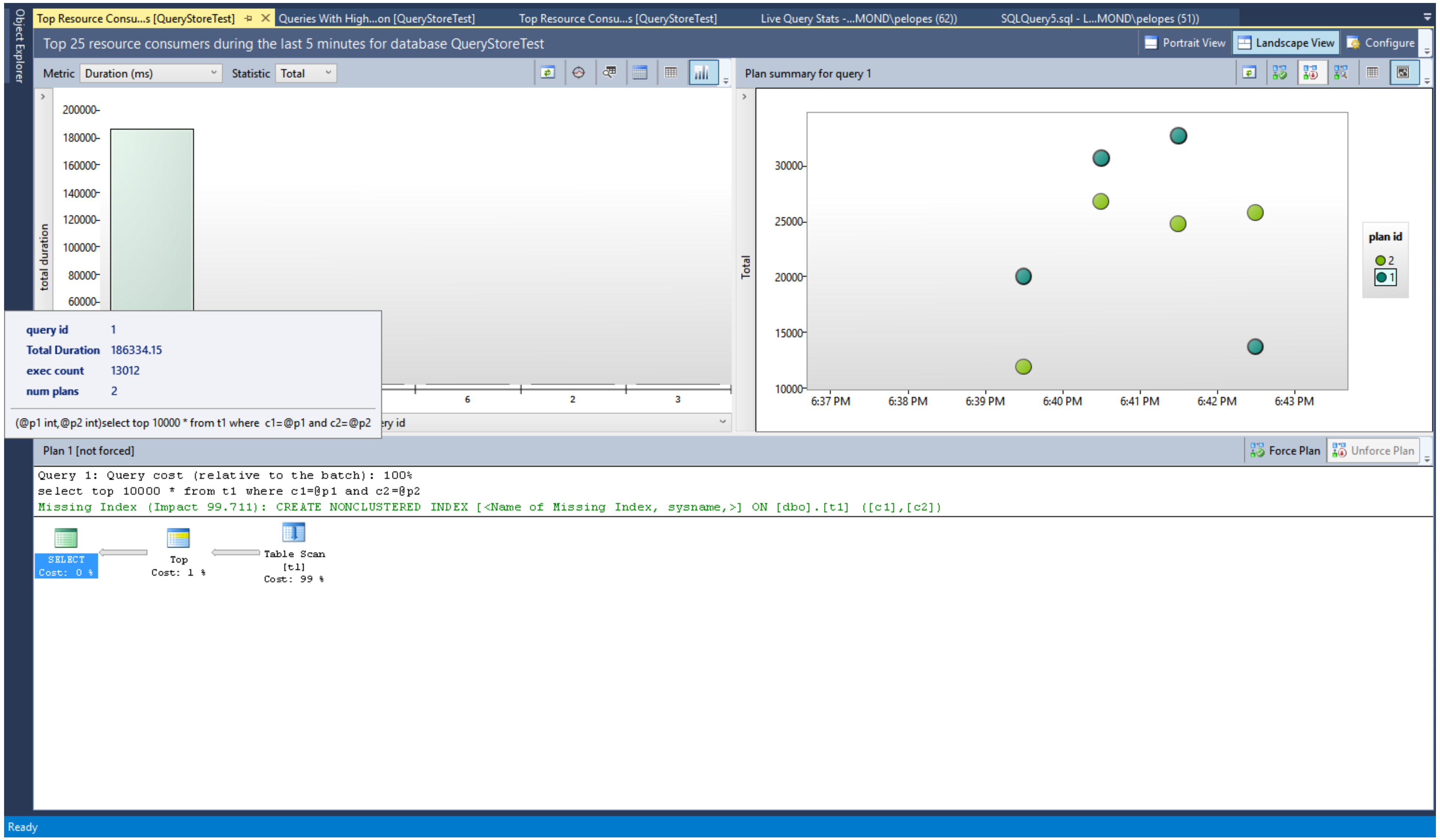
Query performance insights in SSMS

- Still in last v16:
  - Support for multi-statement showplan comparison
  - Per-operator level performance stats in showplan Properties window
  - Query Store
    - Filter by number of different plans
- New with v17:
  - Query Store: new reports
  - Query analysis scenarios: Introduces CE diff search


Query Store
Regressed Queries
Overall Resource Consumption
Top Resource Consuming Queries
Queries With Forced Plans
Queries With High Variation
Tracked Queries

Query Store UI

The query store feature provides DBAs with insight on query plan choice and performance



Top 25 resource consumers during the last 5 minutes for database QueryStoreTest

query id	1
Total Duration	186334.15
exec count	13012
num plans	2

(@p1 int,@p2 int)select top 10000 * from t1 where c1=@p1 and c2=@p2

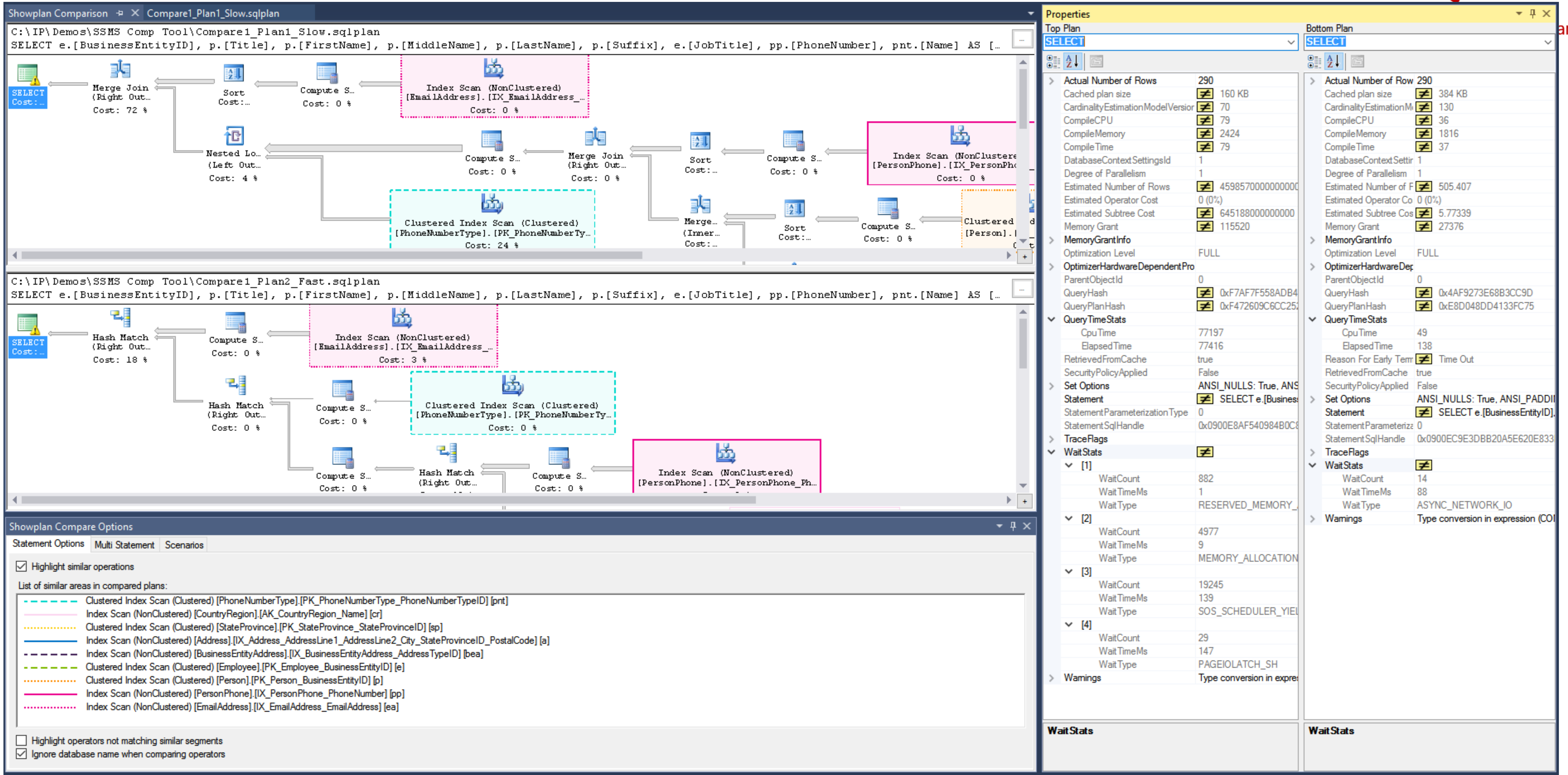
Plan 1 [not forced]

```
Query 1: Query cost (relative to the batch): 100%
select top 10000 * from t1 where c1=@p1 and c2=@p2
Missing Index (Impact 99.711): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[t1] ([c1],[c2])
```

SELECT Cost: 0 % ← Top Cost: 1 % ← Table Scan [t1] Cost: 99 %

SQL Server Tiger Team

SSMS Plan Comparison



Showplan Comparison
Compare1_Plan1_Slow.sqlplan
SELECT Cost:…
Merge Join (Right Out… Cost: 72 %
Sort Cost:…
Compute S… Cost: 0 %
Index Scan (NonClustered) [EmailAddress].[IX_EmailAddress_… Cost: 0 %
Nested Lo… (Left Out… Cost: 4 %
Compute S… Cost: 0 %
Merge Join (Right Out… Cost: 0 %
Sort Cost:…
Compute S… Cost: 0 %
Index Scan (NonClustere [PersonPhone].[IX_PersonPho Cost: 0 %
Clustered Index Scan (Clustered) [PhoneNumberType].[PK_PhoneNumberTy… Cost: 24 %
Merge… (Inner… Cost:…
Sort Cost:…
Compute S… Cost: 0 %
Clustered [Person].
SELECT Cost:…
Hash Match (Right Out… Cost: 18 %
Compute S… Cost: 0 %
Index Scan (NonClustered) [EmailAddress].[IX_EmailAddress_… Cost: 3 %
Hash Match (Right Out… Cost: 0 %
Compute S… Cost: 0 %
Clustered Index Scan (Clustered) [PhoneNumberType].[PK_PhoneNumberTy… Cost: 0 %
Compute S… Cost: 0 %
Hash Match (Right Out…
Compute S… Cost: 0 %
Index Scan (NonClustered) [PersonPhone].[IX_PersonPhone_Ph…
Properties
Showplan Compare Options
Statement Options
Multi Statement
Scenarios

```
C:\IP\Demos\SSMS Comp Tool\Compare1_Plan1_Slow.sqlplan
SELECT e.[BusinessEntityID], p.[Title], p.[FirstName], p.[MiddleName], p.[LastName], p.[Suffix], e.[JobTitle], pp.[PhoneNumber], pnt.[Name] AS […
```

```
C:\IP\Demos\SSMS Comp Tool\Compare1_Plan2_Fast.sqlplan
SELECT e.[BusinessEntityID], p.[Title], p.[FirstName], p.[MiddleName], p.[LastName], p.[Suffix], e.[JobTitle], pp.[PhoneNumber], pnt.[Name] AS […
```

Top Plan

SELECT

Actual Number of Rows		290
Cached plan size	≠	160 KB
CardinalityEstimationModelVersior	≠	70
CompileCPU	≠	79
CompileMemory	≠	2424
CompileTime	≠	79
DatabaseContextSettingsId		1
Degree of Parallelism		1
Estimated Number of Rows	≠	459857000000000
Estimated Operator Cost		0 (0%)
Estimated Subtree Cost	≠	645188000000000
Memory Grant	≠	115520
MemoryGrantInfo		
Optimization Level		FULL
OptimizerHardwareDependentPro		
ParentObjectId		0
QueryHash	≠	0xF7AF7F558ADB4
QueryPlanHash	≠	0xF472609C6CC25
QueryTimeStats		
CpuTime		77197
ElapsedTime		77416
RetrievedFromCache		true
SecurityPolicyApplied		False
Set Options		ANSI_NULLS: True, ANS
Statement	≠	SELECT e.[Busines
StatementParameterizationType		0
StatementSqlHandle		0x0900E8AF540984B0C8
TraceFlags		
WaitStats	≠	
[1]		
WaitCount		882
WaitTimeMs		1
WaitType		RESERVED_MEMORY_
[2]		
WaitCount		4977
WaitTimeMs		9
WaitType		MEMORY_ALLOCATION
[3]		
WaitCount		19245
WaitTimeMs		139
WaitType		SOS_SCHEDULER_YIEL
[4]		
WaitCount		29
WaitTimeMs		147
WaitType		PAGEIOLATCH_SH
Warnings		Type conversion in expres

WaitStats

Bottom Plan

SELECT

Actual Number of Row		290
Cached plan size	≠	384 KB
CardinalityEstimationM	≠	130
CompileCPU	≠	36
CompileMemory	≠	1816
CompileTime	≠	37
DatabaseContextSettir		1
Degree of Parallelism		1
Estimated Number of F	≠	505.407
Estimated Operator Co		0 (0%)
Estimated Subtree Cos	≠	5.77339
Memory Grant	≠	27376
MemoryGrantInfo		
Optimization Level		FULL
OptimizerHardwareDep		
ParentObjectId		0
QueryHash	≠	0x4AF9273E68B3CC9D
QueryPlanHash	≠	0xE8D048DD4133FC75
QueryTimeStats		
CpuTime		49
ElapsedTime		138
Reason For Early Term	≠	Time Out
RetrievedFromCache		true
SecurityPolicyApplied		False
Set Options		ANSI_NULLS: True, ANSI_PADDI
Statement	≠	SELECT e.[BusinessEntityID],
StatementParameteriz		0
StatementSqlHandle		0x0900EC9E3DBB20A5E620E833
TraceFlags		
WaitStats	≠	
WaitCount		14
WaitTimeMs		88
WaitType		ASYNC_NETWORK_IO
Warnings		Type conversion in expression (CO

WaitStats

Showplan Compare Options

☑ Highlight similar operations

List of similar areas in compared plans:

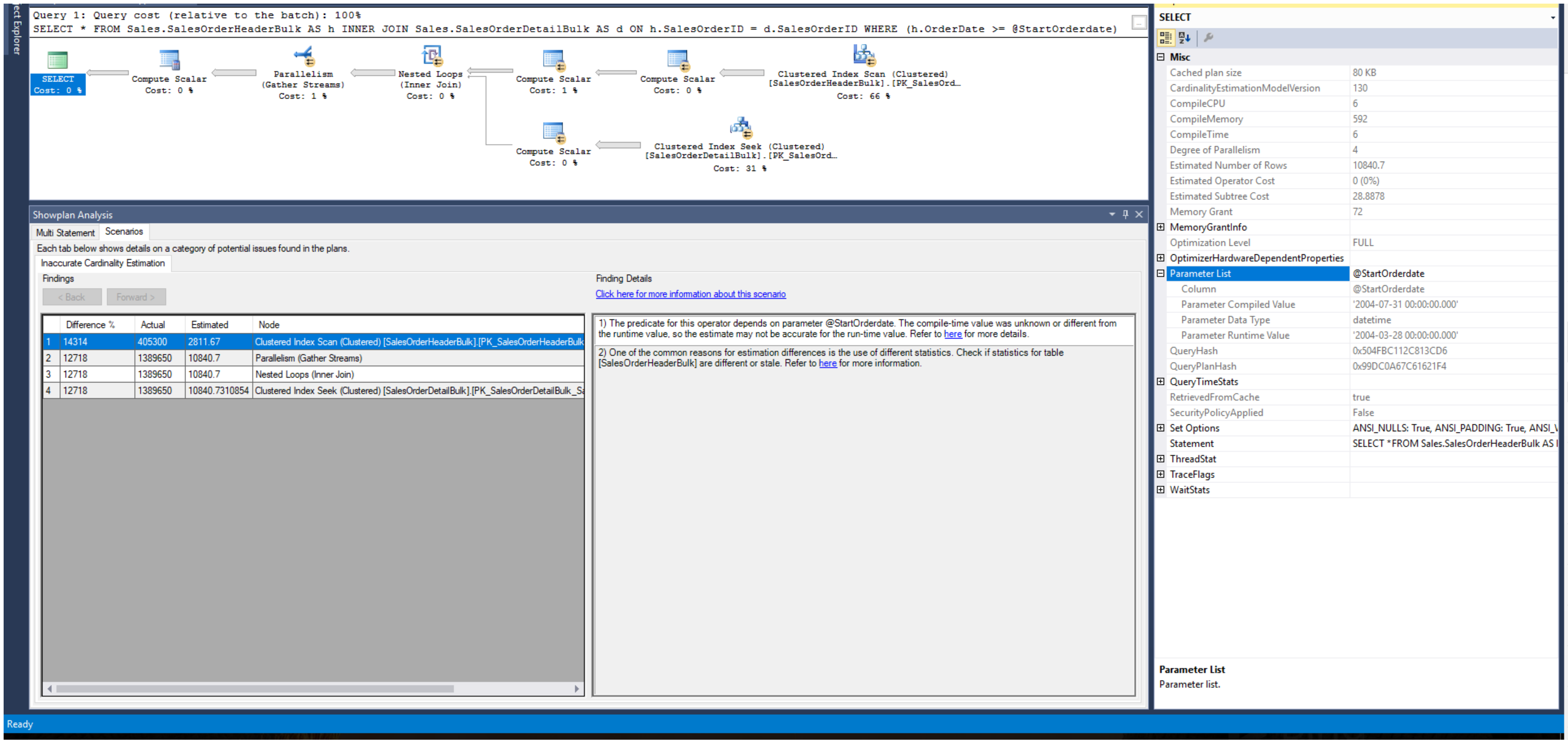
- Clustered Index Scan (Clustered) [PhoneNumberType].[PK_PhoneNumberType_PhoneNumberTypeID] [pnt]
- Index Scan (NonClustered) [CountryRegion].[AK_CountryRegion_Name] [cr]
- Clustered Index Scan (Clustered) [StateProvince].[PK_StateProvince_StateProvinceID] [sp]
- Index Scan (NonClustered) [Address].[IX_Address_AddressLine1_AddressLine2_City_StateProvinceID_PostalCode] [a]
- Index Scan (NonClustered) [BusinessEntityAddress].[IX_BusinessEntityAddress_AddressTypeID] [bea]
- Clustered Index Scan (Clustered) [Employee].[PK_Employee_BusinessEntityID] [e]
- Clustered Index Scan (Clustered) [Person].[PK_Person_BusinessEntityID] [p]
- Index Scan (NonClustered) [PersonPhone].[IX_PersonPhone_PhoneNumber] [pp]
- Index Scan (NonClustered) [EmailAddress].[IX_EmailAddress_EmailAddress] [ea]

☐ Highlight operators not matching similar segments

☑ Ignore database name when comparing operators

SSMS Plan Analysis

er Team



```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Sales.SalesOrderHeaderBulk AS h INNER JOIN Sales.SalesOrderDetailBulk AS d ON h.SalesOrderID = d.SalesOrderID WHERE (h.OrderDate >= @StartOrderdate)
```

Each tab below shows details on a category of potential issues found in the plans.

	Difference %	Actual	Estimated	Node
1	14314	405300	2811.67	Clustered Index Scan (Clustered) [SalesOrderHeaderBulk].[PK_SalesOrderHeaderBulk
2	12718	1389650	10840.7	Parallelism (Gather Streams)
3	12718	1389650	10840.7	Nested Loops (Inner Join)
4	12718	1389650	10840.7310854	Clustered Index Seek (Clustered) [SalesOrderDetailBulk].[PK_SalesOrderDetailBulk_S

Click here for more information about this scenario

1) The predicate for this operator depends on parameter @StartOrderdate. The compile-time value was unknown or different from the runtime value, so the estimate may not be accurate for the run-time value. Refer to here for more details.

2) One of the common reasons for estimation differences is the use of different statistics. Check if statistics for table [SalesOrderHeaderBulk] are different or stale. Refer to here for more information.

Misc	
Cached plan size	80 KB
CardinalityEstimationModelVersion	130
CompileCPU	6
CompileMemory	592
CompileTime	6
Degree of Parallelism	4
Estimated Number of Rows	10840.7
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	28.8878
Memory Grant	72
MemoryGrantInfo	
Optimization Level	FULL
OptimizerHardwareDependentProperties	
Parameter List	@StartOrderdate
Column	@StartOrderdate
Parameter Compiled Value	'2004-07-31 00:00:00.000'
Parameter Data Type	datetime
Parameter Runtime Value	'2004-03-28 00:00:00.000'
QueryHash	0x504FBC112C813CD6
QueryPlanHash	0x99DC0A67C61621F4
QueryTimeStats	
RetrievedFromCache	true
SecurityPolicyApplied	False
Set Options	ANSI_NULLS: True, ANSI_PADDING: True, ANSI_
Statement	SELECT * FROM Sales.SalesOrderHeaderBulk AS
ThreadStat	
TraceFlags	
WaitStats	

Parameter List
Parameter list.

SQL Server Tiger Team

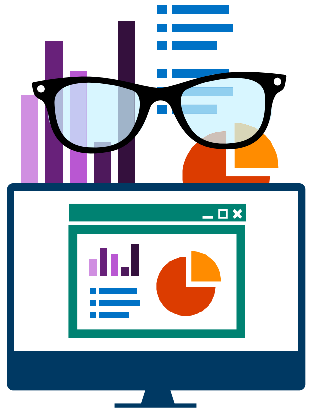
Demo

Query Store UI and Plan Comparison

Plan Analysis

In Review: Objectives And Takeaways

SQL Server Tiger Team



Learn about new diagnostics improvements for SQL Server engine

Learn how to use the new diagnostics to troubleshoot common performance issues

Bookmarks

SQL Server Tiger Team Blog	http://aka.ms/sqlserverteam
Tiger Toolbox GitHub	http://aka.ms/tigertoolbox
SQL Server Release Blog	http://aka.ms/sqlreleases
BP Check	http://aka.ms/bpcheck
SQL Server Standards Support	http://aka.ms/sqlstandards
Trace Flags	http://aka.ms/traceflags
SQL Server Support lifecycle	http://aka.ms/sqllifecycle
SQL Server Updates	http://aka.ms/sqlupdates
Twitter	@mssqltiger

Microsoft