SQLintersection

Session: Wednesday, 10:15am-11:30am

Query Performance Insights
What's new?

Pedro Lopes

SQL intersection

Speaker: Pedro Lopes

- Senior Program Manager
- SQL Server Engineering “Tiger” team owns in-market and vNext of SQL Server
- Focused on SQL Server Relational Engine (Query Processor, Query Perf)
- 9+ years at Microsoft

@SQLPedro

http://aka.ms/sqlserverteam

SQL intersection

Reminder: Intersect with Speakers and Attendees

- **Tweet tips and tricks that you learn and follow tweets posted by your peers!**
  - Follow: #SQLintersection and/or #DEVintersection
- **Join us – Wednesday Evening – for SQLafterDark**
  - Doors open at 7:00 pm
  - Trivia game starts at 7:30 pm

    Winning team receives something fun!
  - Raffle at the end of the night

    Lots of great items to win including a seat in a SQLskills Immersion Event!
  - The first round of drinks is sponsored by SentryOne and SQLskills

SQLafter Dark

SQL intersection

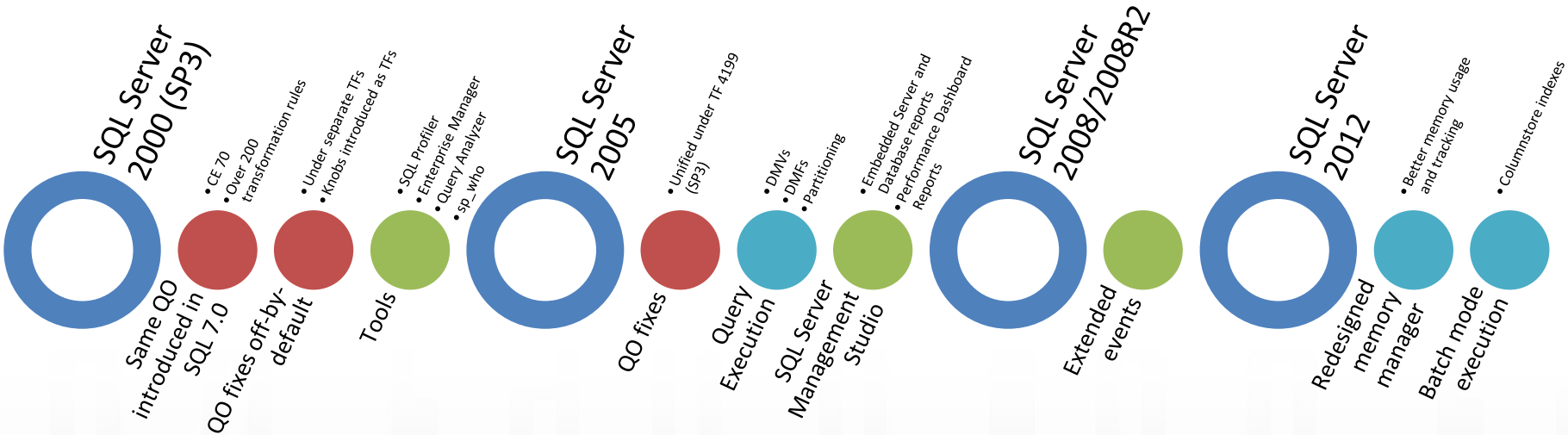
SQLskills immerse yourself in sql server & SentryOne™

Overview

- **A brief history of SQL Server query performance**
- **Diagnostics enhancements**
  - For instance/workload level signals – tools
  - For query performance analysis – plan properties
  - For performance troubleshooting investigations – xEvents + LEP

SQL intersection

Query Performance Journey



Query Performance Journey

Query Optimizer

Diagnostics and Tools

Query Execution

Diagnostics Enhancements

For instance/workload level signals

SQL intersection

Performance Dashboard in SSMS

Microsoft SQL Server Performance Dashboard

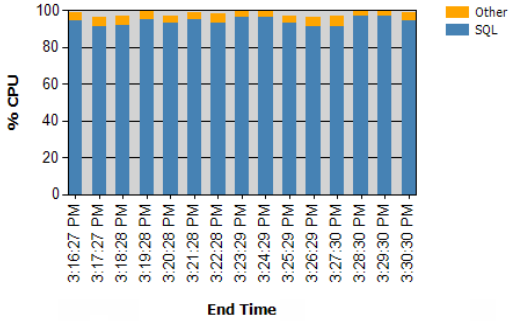
Report Local Time: 5/31/2017 3:31:04 PM

[redacted] (13.0.4422.0 - Enterprise Edition (64-bit))

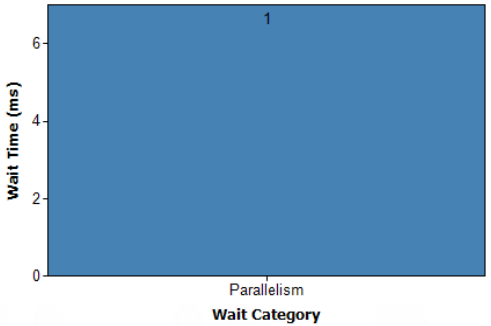
Overall performance may be degraded because the system shows signs of being CPU-bound.This SQL Server instance is consuming the majority of the CPU. Click on any of the SQL data points in the chart below to investigate further.

Starting with **SSMS v17.2**
No extra downloads!
No new schema to deploy!

System CPU Utilization



Current Waiting Requests



Current Activity

	User Requests	User Sessions
Count	27	32
Elapsed Time (ms)	4573004	741818
CPU Time (ms)	2043203(44.68%)	101108(13.63%)
Wait Time (ms)	2529801(55.32%)	640710(86.37%)
Cache Hit Ratio	100.000%	98.313%

Historical Information

Waits | IO Statistics
Latches

Expensive Queries

By CPU | By Duration
By Logical Reads | By Physical Reads
By Logical Writes | By CLR Time

Miscellaneous Information

Active Traces	1
Active Xevent Sessions	4
Databases	16
Missing Indexes	11

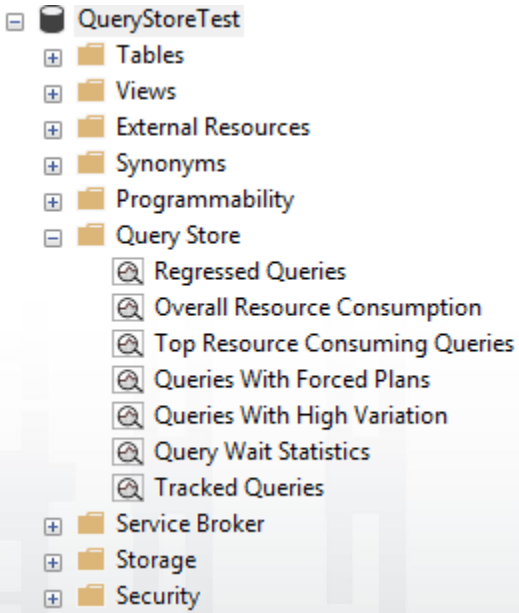
Categorized Wait stats page

New categorized Latches page

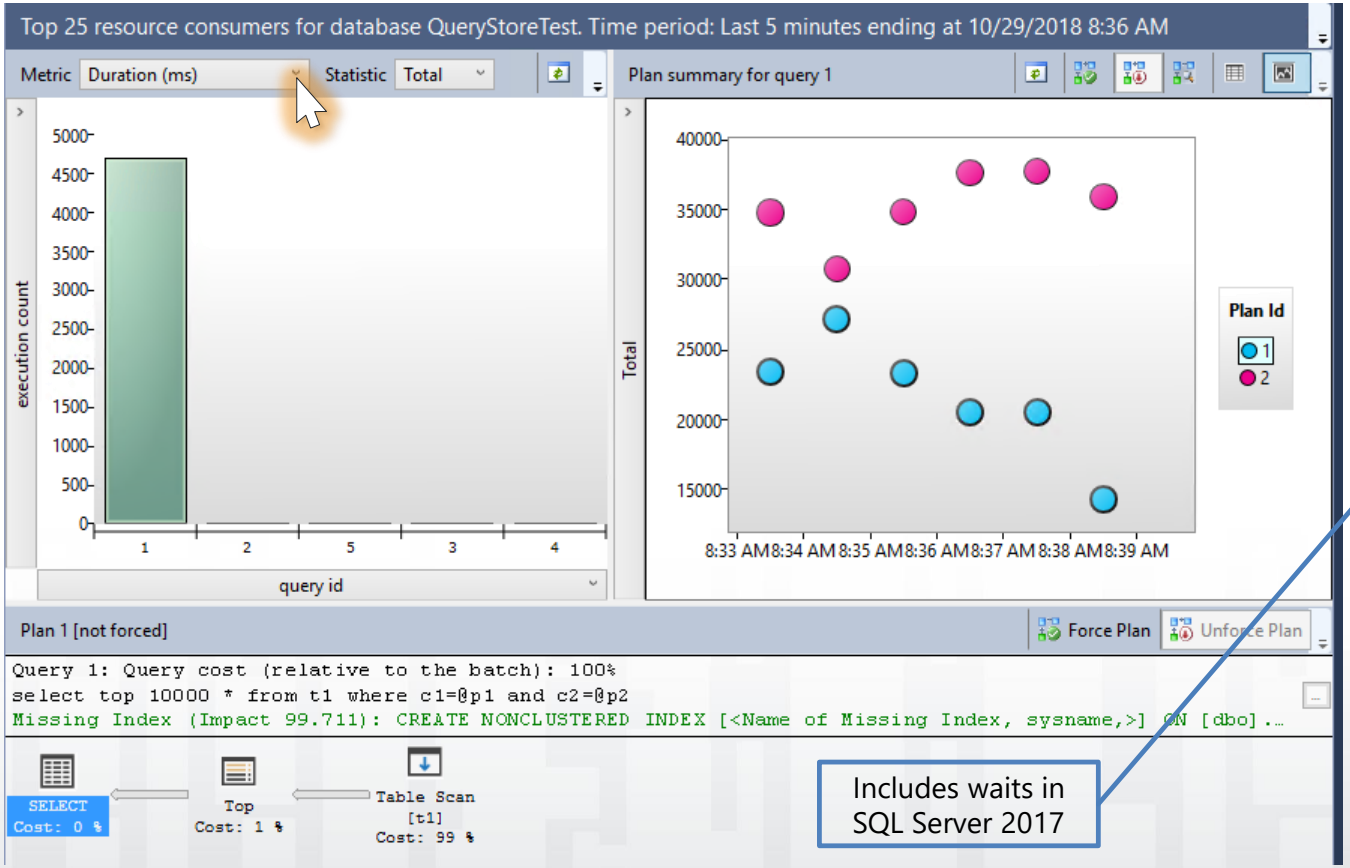
Scoring added to Missing Index Report

Query Store

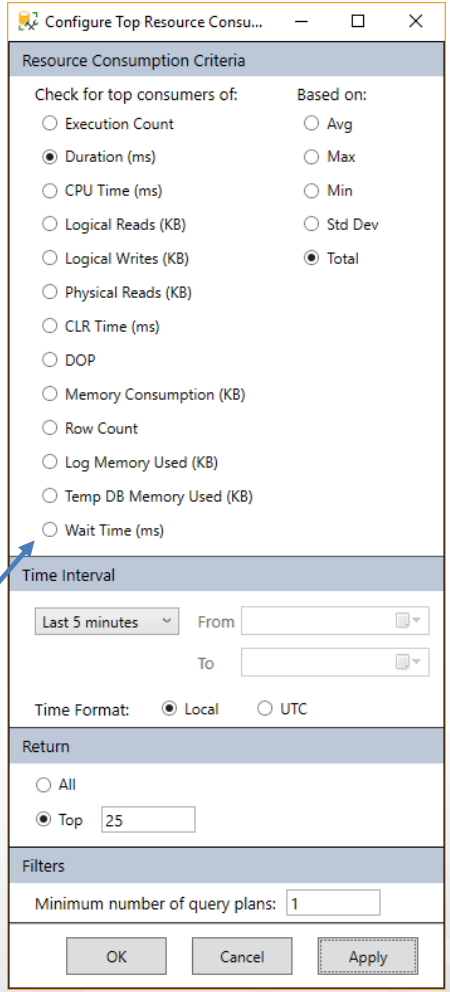
Comprehensive query-performance information when you need it most!



Query Store – Top Consumers

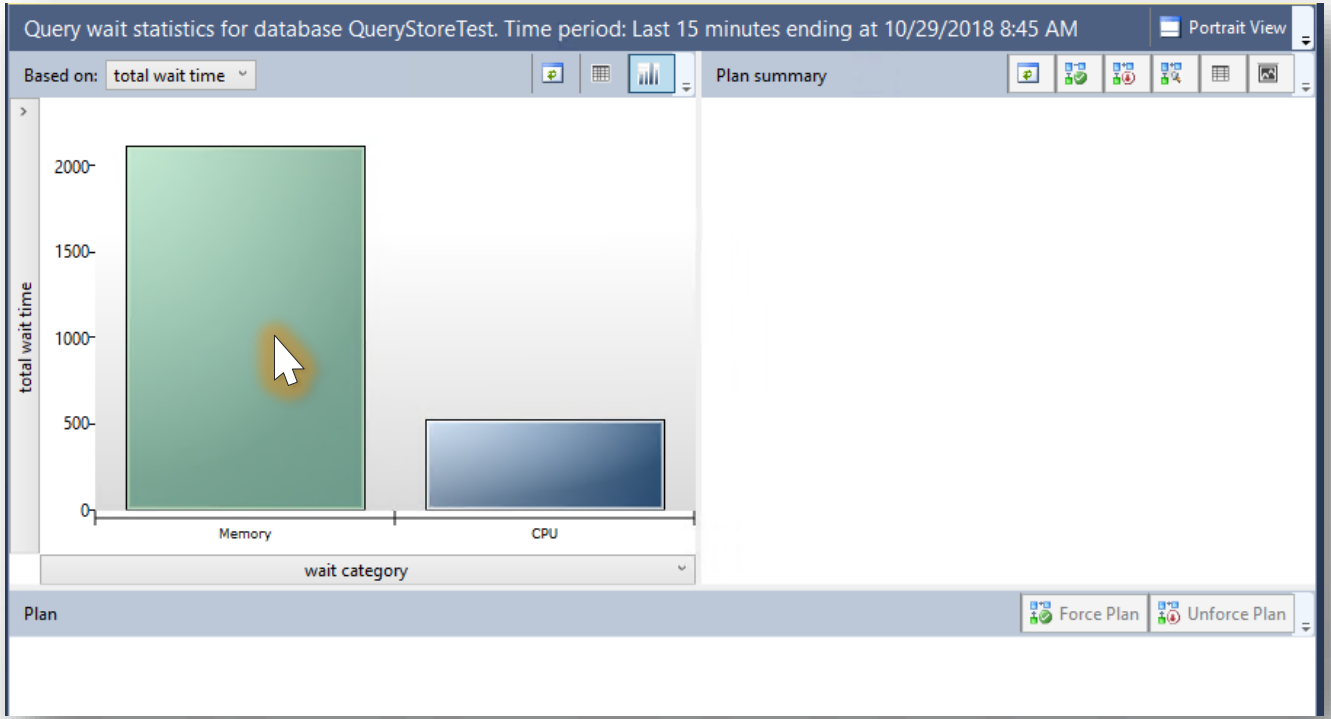


Includes waits in SQL Server 2017



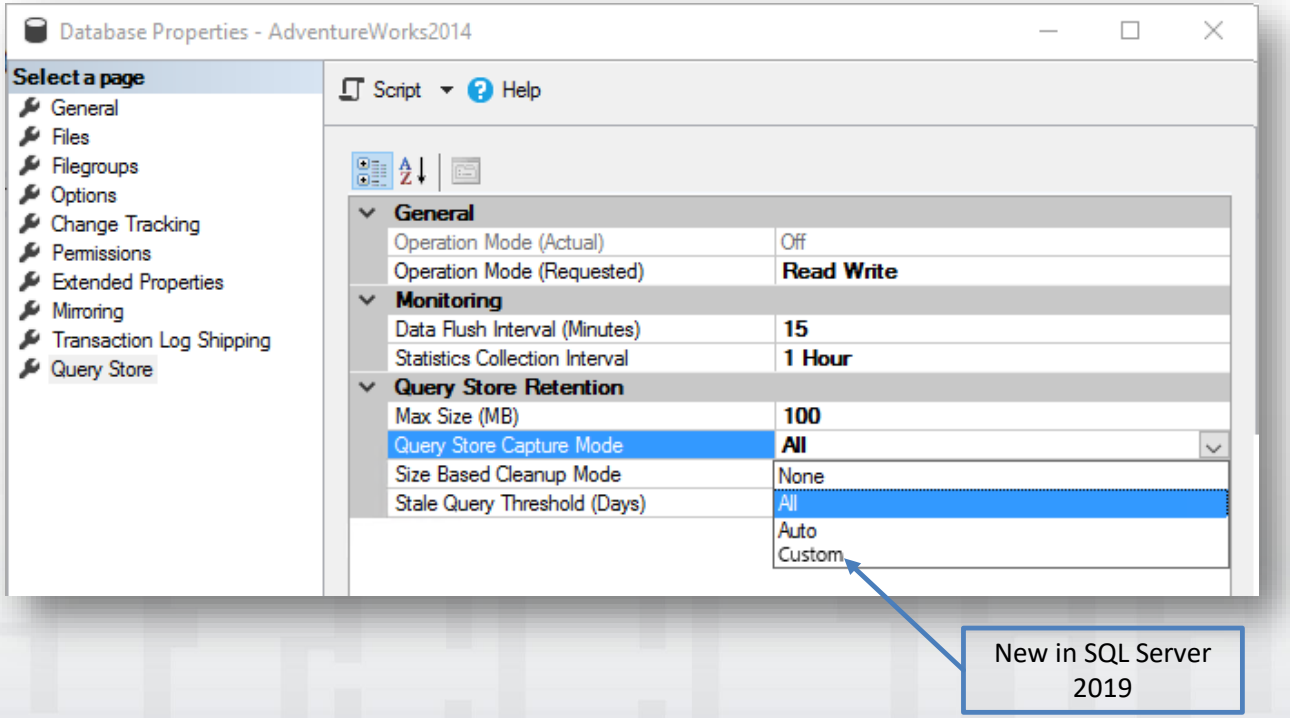
Query Store – Wait Categories

And now waits-based troubleshooting is also available in UI…

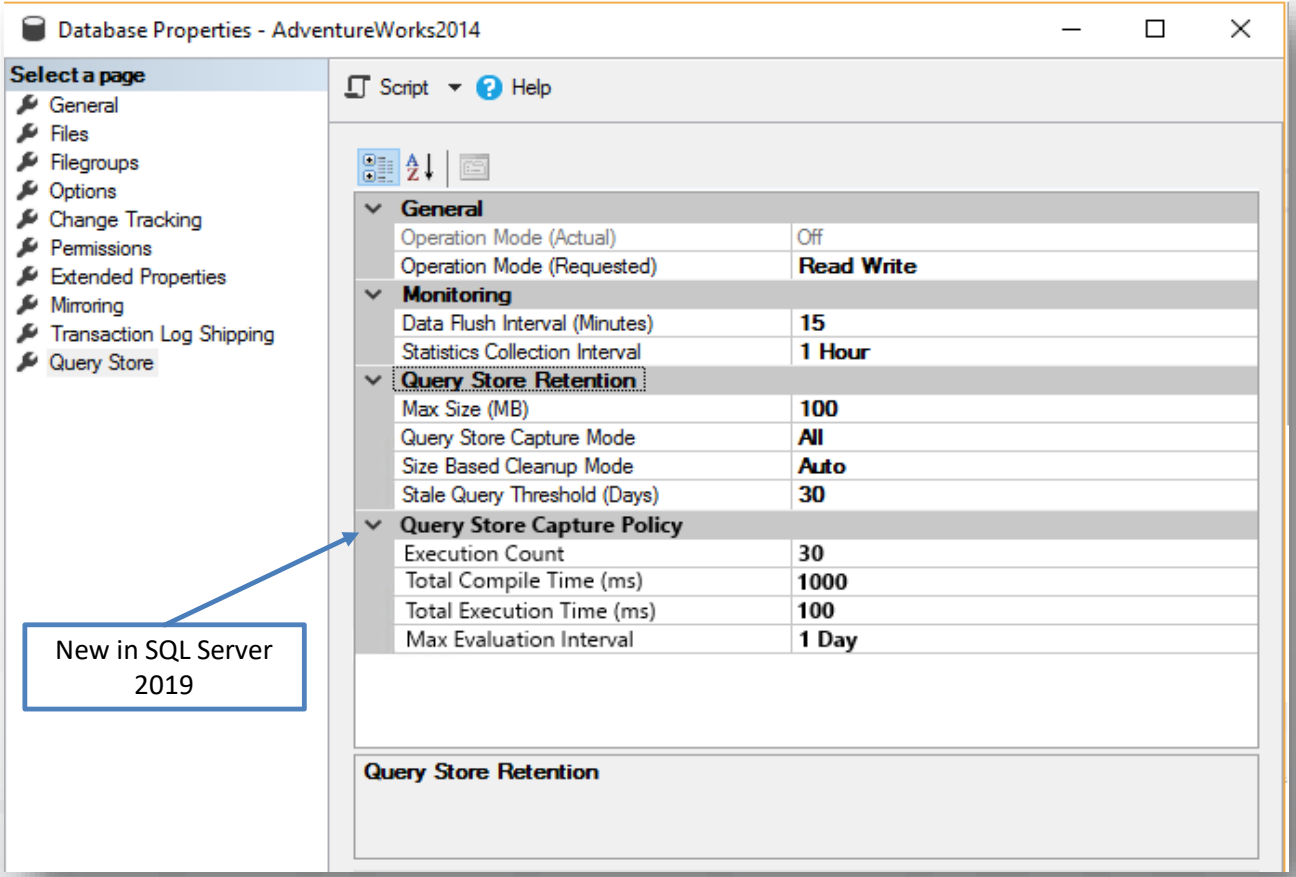


New for SSMS 18

Query Store - Configurations



Query Store - Configurations



Diagnostics Improvements

For query performance analysis

SQL intersection

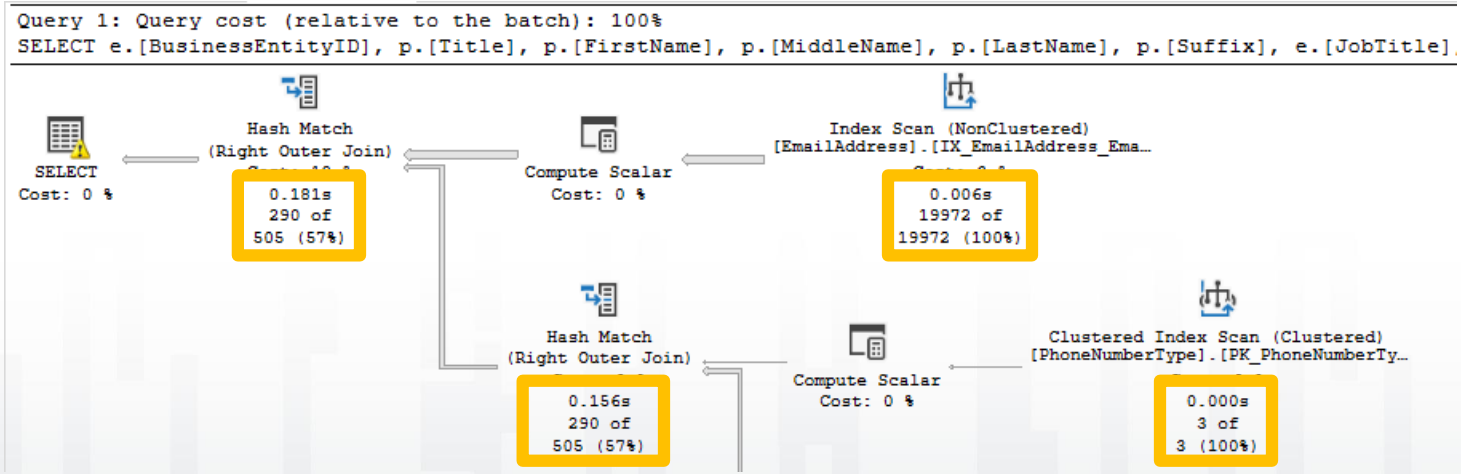
Query plans: fundamental diagnostic map

- 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

SQL intersection

Faster identification of heavy nodes

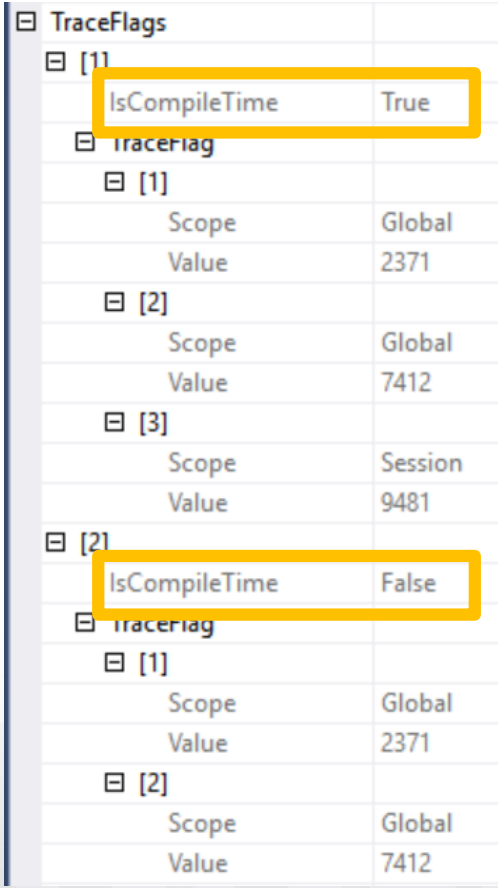
- **SSMS v18 showplan surfaces information on:**
  - Elapsed time per operator that consumes data
  - <actual rows> of <estimated rows> (percent of estimate)



Note: even on an Actual execution plan, the Cost Pct is based on estimations. This is <u>not</u> an accurate measure of true operator cost.

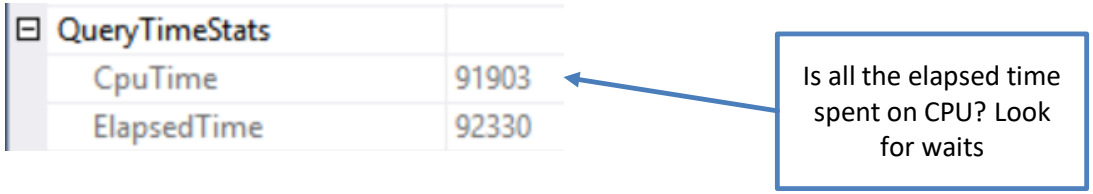
Getting all context info in Showplan: Trace Flags

- **Shows list of active trace flags:**
  - Query
  - Session
  - Global
- **Useful to understand if active Trace Flags influence execution context**
- **IsCompileTime = True**
  - Were present when plan was created and cached
- **IsCompileTime = False**
  - Where not present at plan execution time

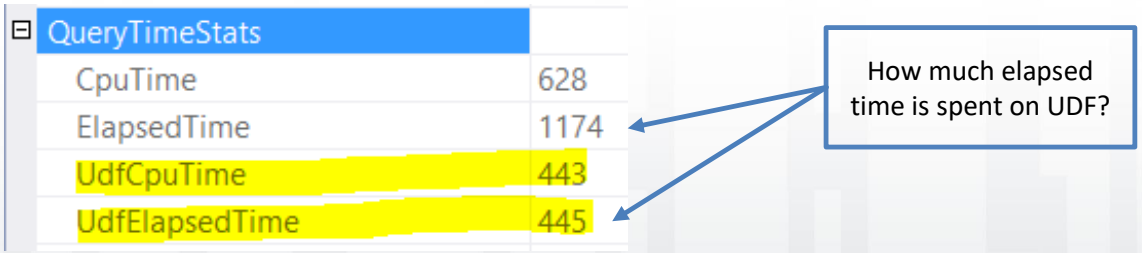


Getting all context info in Showplan: Times

- **Persisting information on elapsed and CPU times**



- **And Scalar UDF elapsed and CPU times**



Getting all context info in Showplan: Waits

- **Shows top 10 waits from sys.dm_exec_session_wait_stats**

Correlate waits with overall query times

QueryTimeStats	
CpuTime	1045
ElapsedTime	3010

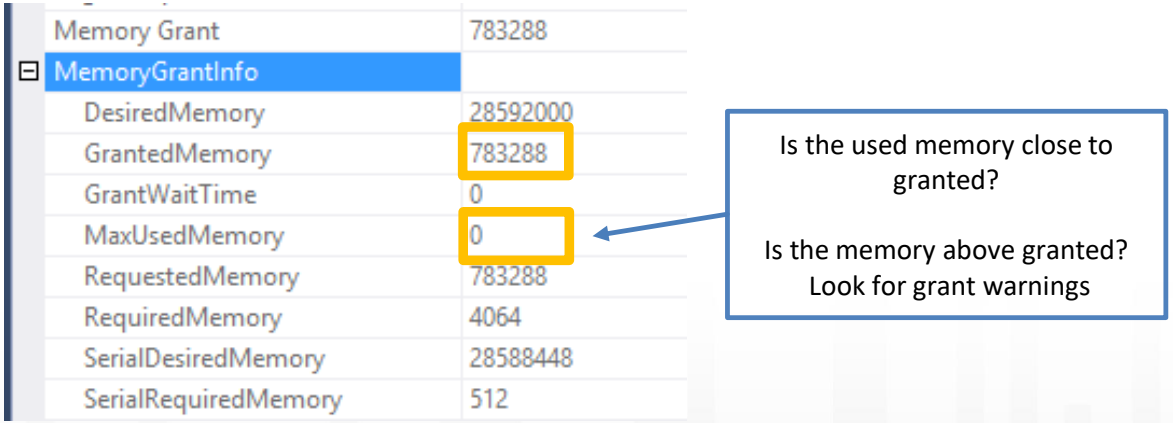
Misc	
Cached plan size	160 KB
CardinalityEstimationModelV	130
CompileCPU	11
CompileMemory	728
CompileTime	136
DatabaseContextSettingsId	3
Degree of Parallelism	12
Estimated Number of Rows	121308
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	4.48002
Memory Grant	80448
MemoryGrantInfo	
Optimization Level	FULL

WaitStats	
[1]	
WaitCount	98
WaitTimeMs	3
WaitType	LATCH_SH
[2]	
WaitCount	50
WaitTimeMs	761
WaitType	PAGEIOLATCH_SH
[3]	
WaitCount	67
WaitTimeMs	1942
WaitType	LATCH_EX
[4]	
WaitCount	129
WaitTimeMs	2509
WaitType	ASYNC_NETWORK_IO
[5]	
WaitCount	2220
WaitTimeMs	30622
WaitType	CXPACKET

Note: Parallelism waits available in SQL Server 2017 CU3 and 2016 SP2;
In ASC order up to SQL Server 2017; DESC order in SQL Server 2019

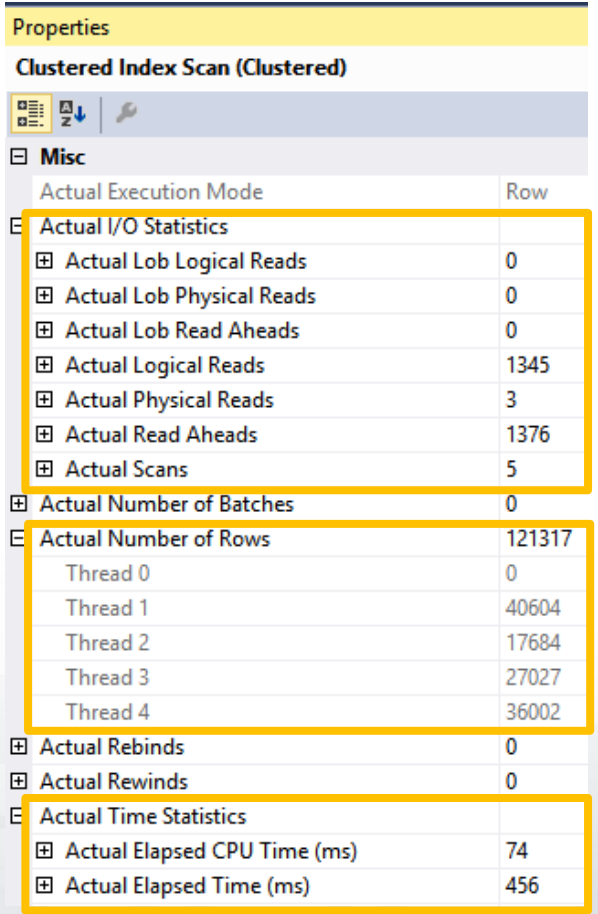
Getting all context info in Showplan: memory

- **Showplan extended to include grant usage per thread and iterator**



- **Also found in sys.dm_exec_query_stats**

Insights into every query plan node



SET STATISTICS IO not needed

SET STATISTICS TIME not needed

Diagnostics Enhancements

For performance troubleshooting investigations

SQL intersection

Problem #1 – “It’s slow”

Defining the problem

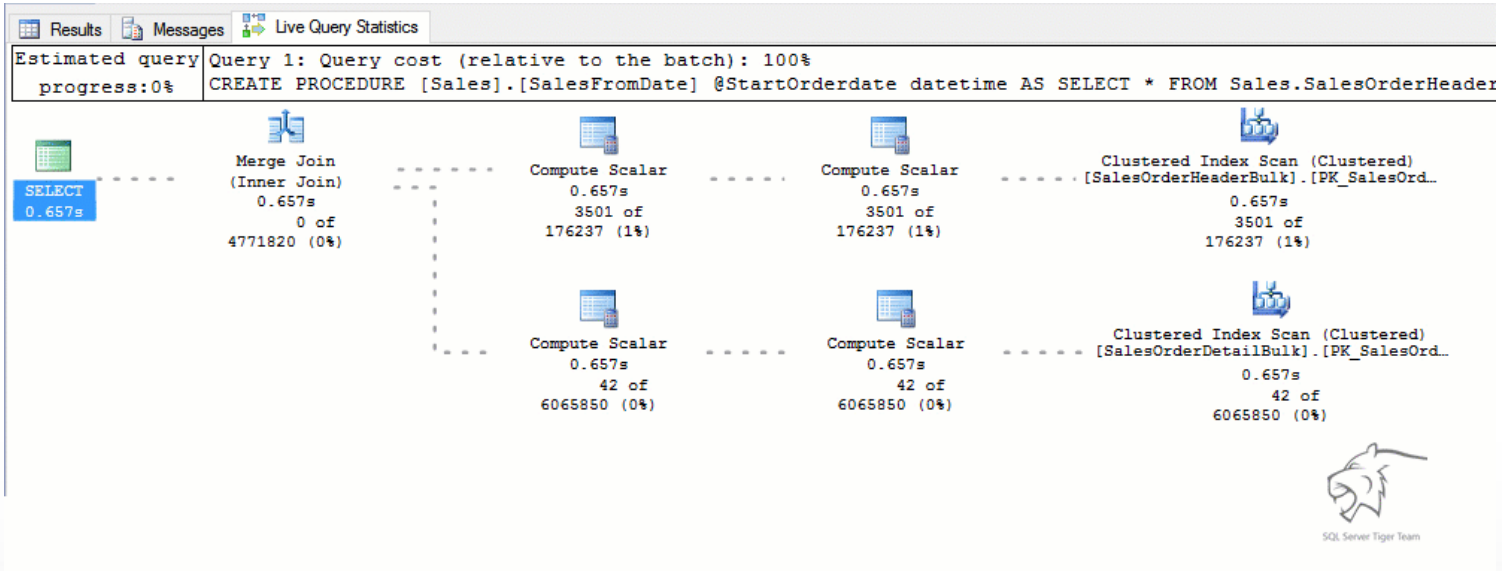
Reasonable hypothesis: a long running query…

Query completion is a prerequisite for an actual query plan

Actual query plans unsuitable for troubleshooting complex performance issues:

- **Long running queries**
- **Queries that run indefinitely and never finish execution.**

What if I could do live query troubleshooting?

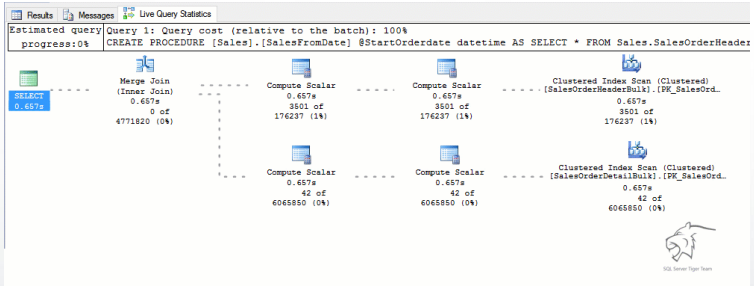


What if I could do live query troubleshooting?

Ok, but to have in-flight query execution visibility, the *query execution statistics profile infrastructure* must be enabled on demand

…its overhead goes up to 75% with TPC-C like workload

It can make bad performance issues worse, so we don't run it all the time…

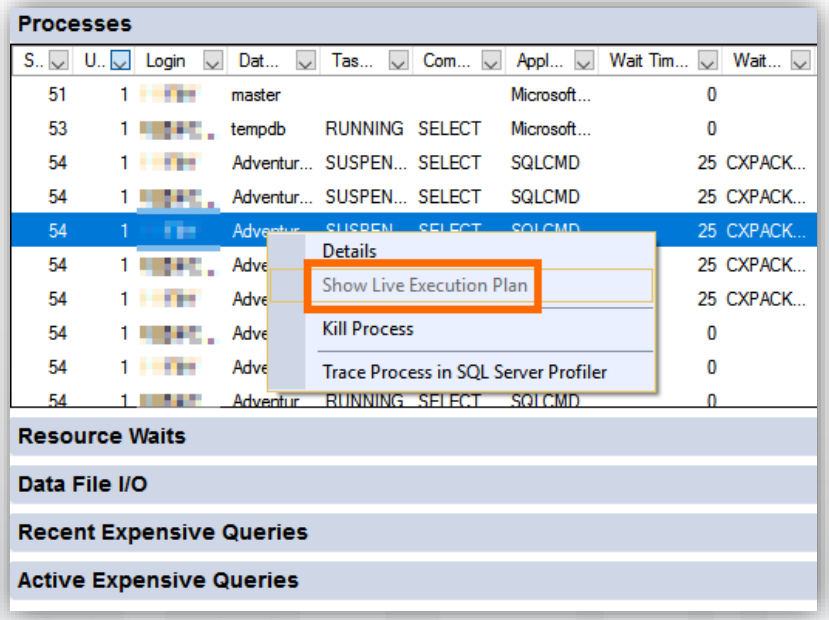


Query progress – anytime, anywhere

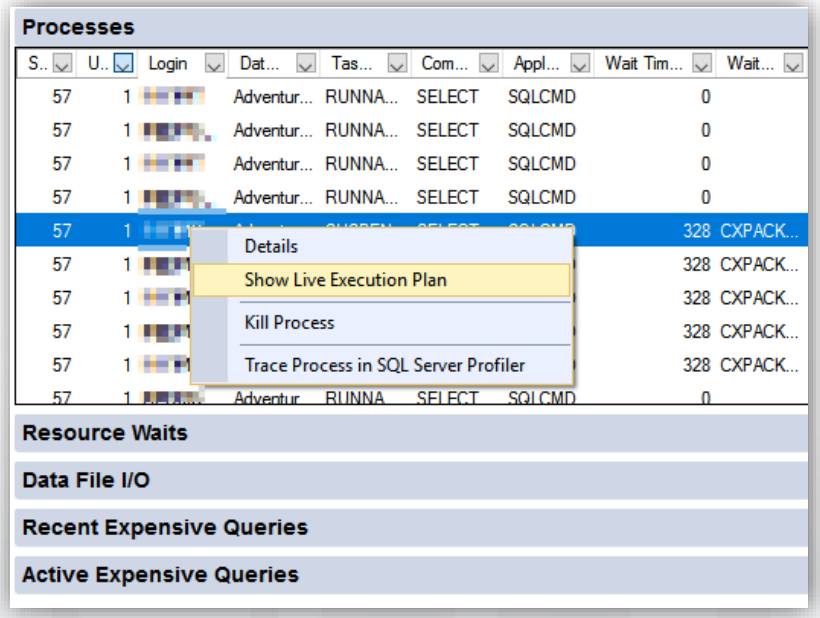
- **Starting with SQL Server 2016 SP1* and 2017, the new *lightweight query execution statistics profile infrastructure* (LWP) allows continuous collection of per-operator query execution statistics**
  - Using global TF 7412
  - Enabling *query_thread_profile* and *query_plan_profile* extended event
    - Or *query_post_execution_plan_profile* extended event in SQL Server 2019
  - Using query hint USE HINT('query_plan_profile') in SQL Server 2017 CU11 and 2016 SP2 CU3 (KB 4458593)
- **Lightweight profiling is ON by default in SQL Server 2019 and no TF needed**

Using SSMS Resource Monitor for live troubleshooting

Without Profiling



With Profiling



Demo

Troubleshooting Long running queries

SQL intersection

What infra is enabled and when?

	Standard Profiling
Globally	XEvent session with query_post_execution_showplan XE; Starting with SQL Server 2012
	Showplan XML trace event in SQL Trace and SQL Server Profiler; Starting with SQL Server 2000
	–

What infra is enabled and when?

	Standard Profiling	Lightweight Profiling
Globally	XEvent session with query_post_execution_showplan XE; Starting with SQL Server 2012	Trace Flag 7412; Starting with SQL Server 2016 SP1
	Showplan XML trace event in SQL Trace and SQL Server Profiler; Starting with SQL Server 2000	XEvent session with query_thread_profile XE; Starting with SQL Server 2014 SP2
	–	XEvent session with query_post_execution_plan_profile XE; Starting with SQL Server 2019

What infra is enabled and when?

	Standard Profiling	Lightweight Profiling
Globally	XEvent session with query_post_execution_showplan XE; Starting with SQL Server 2012	Trace Flag 7412; Starting with SQL Server 2016 SP1
	Showplan XML trace event in SQL Trace and SQL Server Profiler; Starting with SQL Server 2000	XEvent session with query_thread_profile XE; Starting with SQL Server 2014 SP2
	–	XEvent session with query_post_execution_plan_profile XE; Starting with SQL Server 2019
Single session	Use SET STATISTICS XML ON; Starting with SQL Server 2000	
	Use SET STATISTICS PROFILE ON; Starting with SQL Server 2000	
	Click LQS button in SSMS; Starting with SQL Server 2014 SP2	

What infra is enabled and when?

	Standard Profiling	Lightweight Profiling
Globally	XEvent session with query_post_execution_showplan XE; Starting with SQL Server 2012	Trace Flag 7412; Starting with SQL Server 2016 SP1
	Showplan XML trace event in SQL Trace and SQL Server Profiler; Starting with SQL Server 2000	XEvent session with query_thread_profile XE; Starting with SQL Server 2014 SP2
	–	XEvent session with query_post_execution_plan_profile XE; Starting with SQL Server 2019
Single session	Use SET STATISTICS XML ON; Starting with SQL Server 2000	QUERY_PLAN_PROFILE query hint + XEvent session with query_plan_profile XE; Starting with SQL Server 2016 SP2 CU3 and 2017 CU11
	Use SET STATISTICS PROFILE ON; Starting with SQL Server 2000	–
	Click LQS button in SSMS; Starting with SQL Server 2014 SP2	–

Problem #2 – “It’s fine on my end”

Defining the problem

Reasonable hypothesis: a problem query that I cannot repro in "my SSMS"

Getting the actual plan for the production server query plan is needed. But how?

Could I use Query Store? Perhaps, but it collects time aggregates. For some critical scenarios it may not be optimal when you need the singleton plan

What if I could always access the equivalent of last actual execution plan for any query?

In SQL Server 2019 (CTP 2.4), I can!

- **Uses LWP and access the plan through sys.dm_exec_query_plan_stats**
- **It's opt-in:**
  - Trace flag 2451
  - LAST_QUERY_PLAN_STATS database scoped configuration (CTP 2.5)

Demo

Using new query plan xEvents + Last Execution Query Profile

SQL intersection

Review

- **We've covered**
  - What tools we can use to understand the types of workload problems we may face
  - Scenarios that are made easier with lightweight profiling
    - For long-running queries
    - For critical volatile queries that run fine in my test, but slow for the app
  - Just a few of the query plan properties that can reveal insights into query performance, helping us understand the root cause of issues

Tiger Toolbox

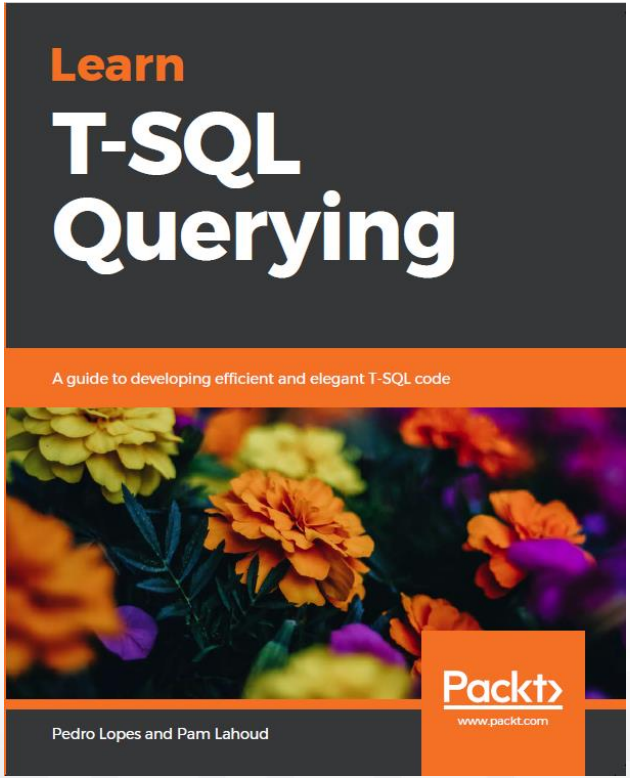
The SQL Server team shares a number of scripts and tools you can (arguably should) use every day in your SQL Server estate:

- Adaptive Index Defrag
- BPCheck (SQL Best Practices and Performance checks)
- Fixing transaction log VLF count
- Maintenance solution that includes integrity checking
- Waits and Latches information
- usp_WhatsUp (Running and Blocked processes, Execution stats)
- System Health Session PowerBI
- Various other support scripts

Find these and more at http://aka.ms/TigerToolbox

References

- One bookmark to rule them all
  https://aka.ms/sqlshortcuts
- @mssqltiger
- Continue learning with our new book!
  https://aka.ms/LearnTSQLQuerying



SQL intersection

Questions?

Microsoft

Don’t forget to complete an online evaluation!

Query Performance Insights – What’s new?

Your evaluation helps organizers build better conferences and helps speakers improve their sessions.

SQL *intersection*

Thank you!

Save the Date

www.SQLintersection.com

Week of November 18, 2019

We're back in Vegas baby!

SQL intersection