Query Store, The Good, Bad, and Ugly

Anat Dror and Jason Hall

Quest

Anat Dror

SQL Server Expert, Quest

SQL Server and DB2 domain expert with over 20 years of experience in a long list of IT related roles. Worked with SQL Server since version 6.5. has a broad and deep understanding of cloud computing, virtualization, database development and administration, performance management and storage. Currently employed as subject matter expert bringing Quest Database Performance Management solutions to life.

/ anat-dror-4521134 @anatdror72 anat.dror@quest.com

Quest

Jason Hall

SQL Server Expert, Quest

Database Systems Consultant specializing in assisting customers monitor, manage, and protect their SQL Server environments. Worked with SQL Server since 2000, and has a broad understanding of SQL Server HADR, Performance Tuning, and infrastructure management.

/jasonfhall @jasonfhall Jason.hall@quest.com

Quest

Agenda

- Introduction
  - Pre SQL Server 2016
  - Overview of Query Store
- The Good
  - What are the benefits of Query Store
  - How can the Query Store be used to solve real world problems
- The Bad
  - Are there any downsides to Query Store
- The Golden Solution
  - Best practices for managing the Query Store
  - Be a hero with the Query Store
- Spotlight Developer
  - FREE performance monitoring and tuning

Confidential

Available Scripts

- For a complete version of all scripts referenced in this presentation, please visit the link below:
  - https://www.quest.com/community/b/en/posts/sql-server-2016-query-store-the-good-the-bad-and-the-ugly

Confidential

Query Store Overview

Quest

Before SQL Server 2016

How do we understand query performance?

- Traces:
  - Must be started and stopped
  - Can cause extremely high overhead
- Extended Events:
  - Considerably more events to track, but same limitations as trace
- DMV's: sys.dm_exec_query_stats, sys.dm_exec_cached_plans, etc…
  - Not organized by time
  - Most DMV's are either real time only, or since the last restart
  - Data is flushed when SQL is restarted
    - Cluster failover…

Confidential

Simplify Performance Tuning: Query Store

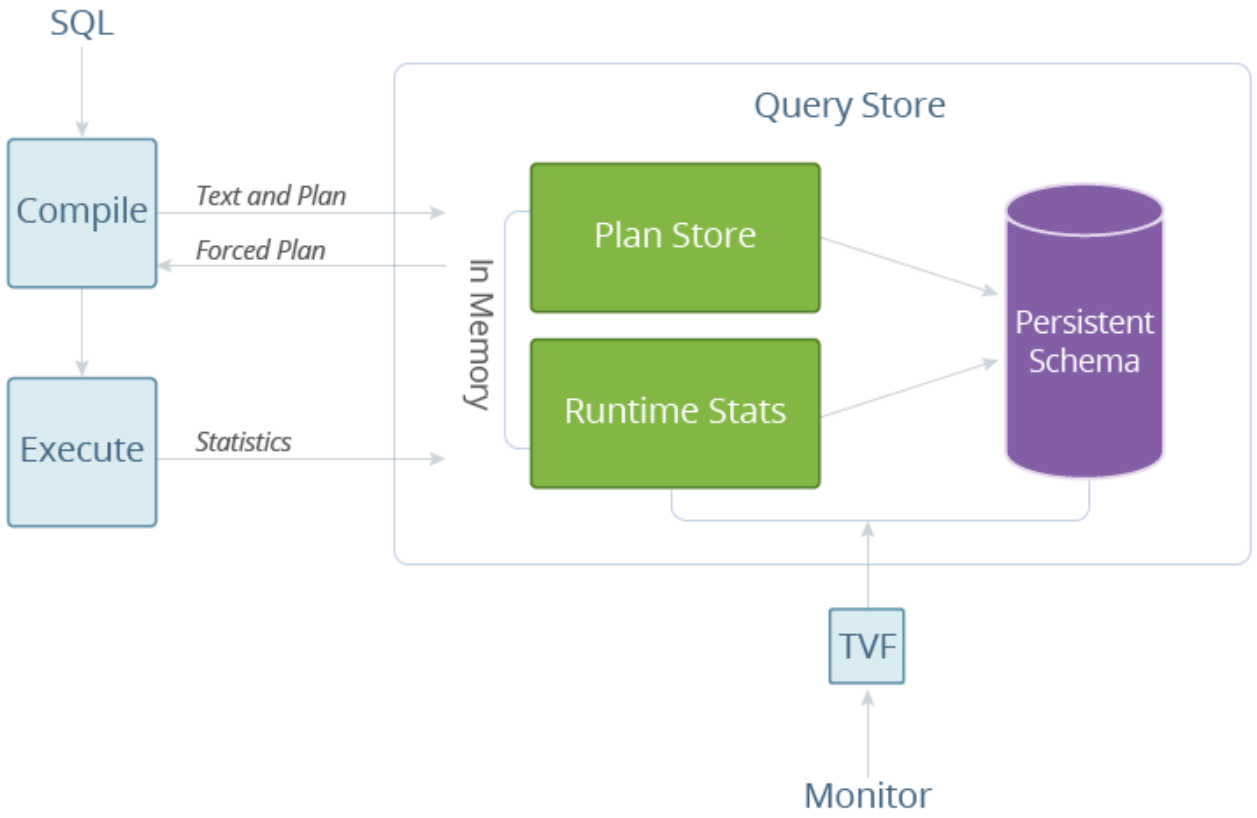
- New in SQL Server 2016

- Automatically captures a history of queries, plans, and runtime statistics
- Performance data is retained during a restart
- Information is aggregated over different time intervals
- Works for natively compiled procedures and in-memory OLTP queries

- Supported on all SQL Server editions
- Enabled at database level

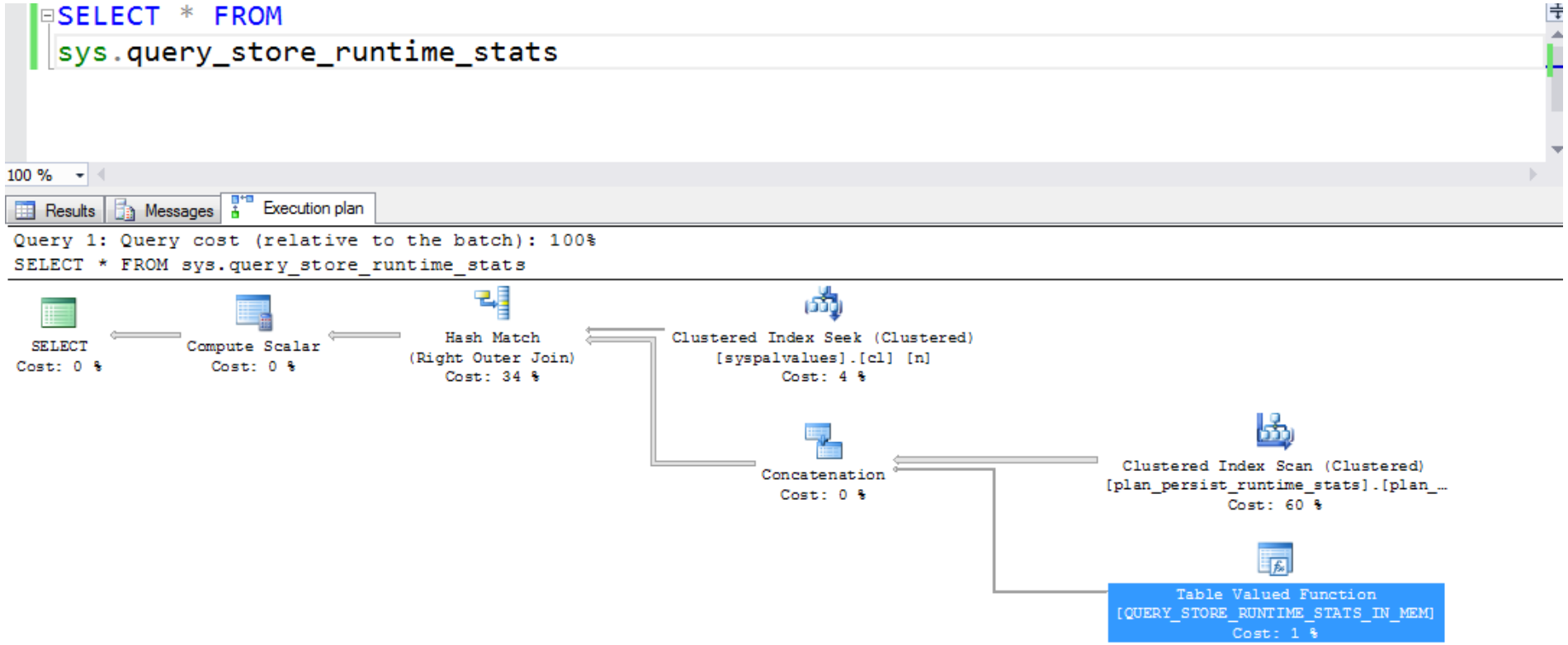
Confidential

Query Store Architecture



Confidential
Quest

Simple Query Store Query



Confidential

Quest

Query Store (New) vs. Query Stats (Old)

Query Store	Query Stats
• Statement level • Query text given • Retain over time and restarts • Aggregate by time windows • Works for in-memory OLTP and disk-based workloads	• Batch level • Query text derived from batch text • Prune to memory shortage and restarts • No time window. Data is aggregated per batch, query and plan since it is logged and until it is evicted. • No information about in-memory OLTP without enabling sys.sp_xtp_control_query_exec_stats (Transact-SQL).

Confidential

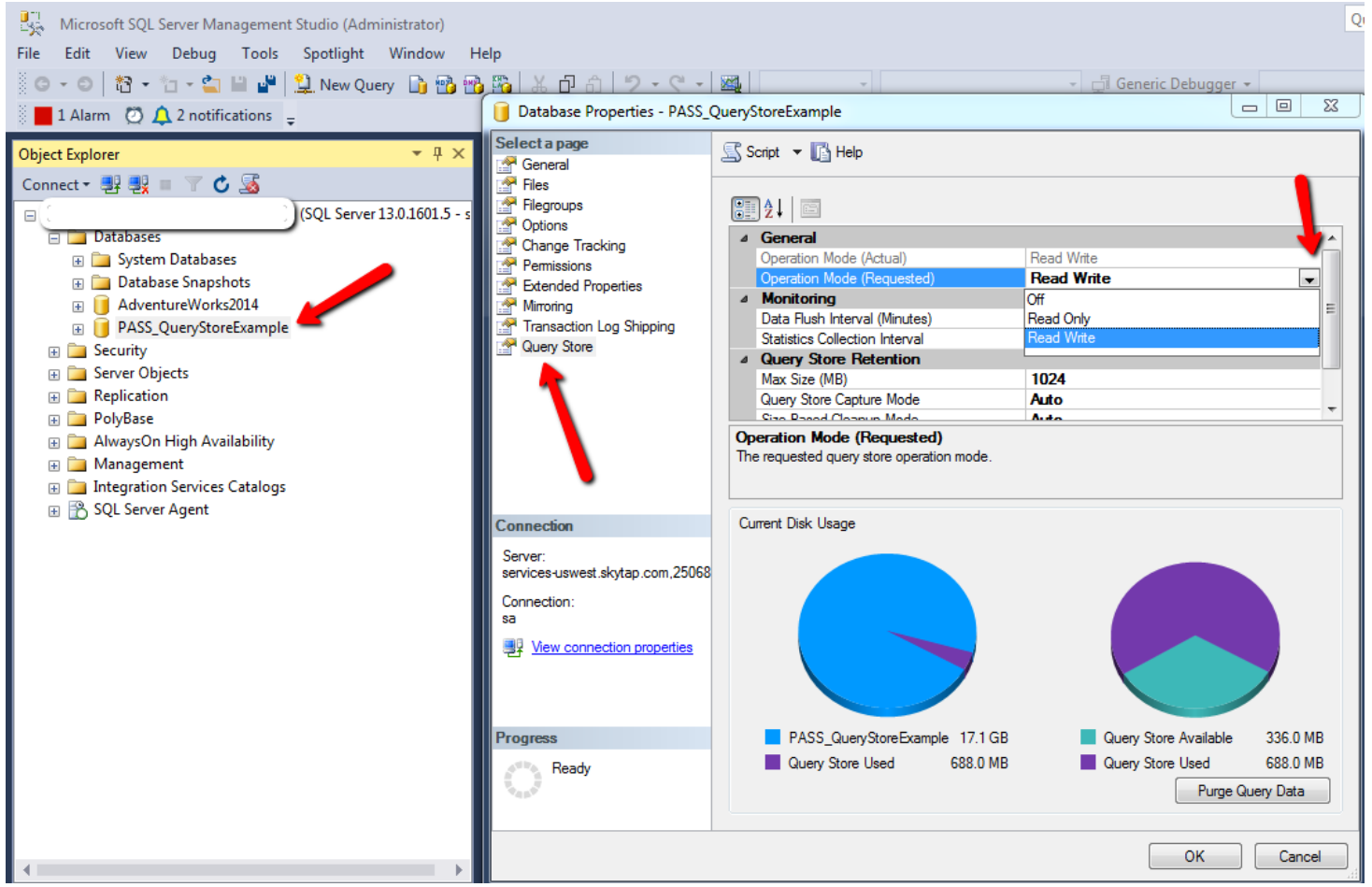
Enable Query Store With TSQL

```
-- Enable Query Store on the database
ALTER DATABASE <database name>
SET QUERY_STORE = ON
GO

-- Configure the Query Store
ALTER DATABASE <database name> SET QUERY_STORE
(
OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30), --number of days to retain data in the query store
DATA_FLUSH_INTERVAL_SECONDS = 900,                  --frequency at which data is persisted to disk
INTERVAL_LENGTH_MINUTES = 60,                       --interval to aggregate runtime execution statistics
MAX_STORAGE_SIZE_MB = 1024,                         --maximum size of the query store
QUERY_CAPTURE_MODE = ALL,                           --type of queries Query Store captures (all, relevant, tracked)
MAX_PLANS_PER_QUERY=5,                              --maximum number of plans maintained for each query.
SIZE_BASED_CLEANUP_MODE = AUTO                      --Controls the cleanup process
)
GO
```


Confidential
Quest™

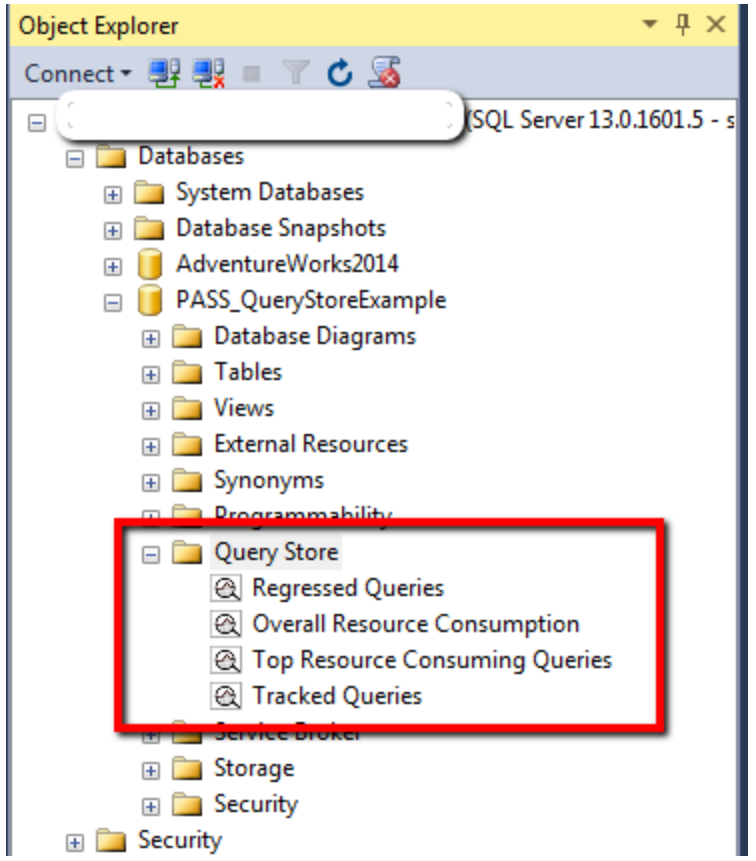
Enable Query Store with SSMS



Confidential

Quest

Out of the Box Reports



Confidential

Quest

Query Store Configuration Parameters

- OPERATION_MODE
  - Read only or Read/Write
- MAX_STORAGE_SIZE_MB
  - Maximum amount of space the Query Store will consume
  - When this is hit, turns read only
- INTERVAL_LENGTH_MINUTES
  - Metrics will be aggregated on this interval (default 60)
- DATA_FLUSH_INTERVAL_SECONDS
  - How often data is written to disk
- QUERY_CAPTURE MODE
  - What queries to capture (ALL, AUTO, NONE)

Confidential

The Good
Benefits of Query Store

Quest

Query Store Use Cases

- View performance metrics broken down by time at query, plan and database level
- Audit history of query plans. Capture ALL changes to query plans.
- If a query is using a bad plan, can force it to use a better one
- Provide pre-customized views geared to support the following scenarios:
  - Find Regressed Queries
  - Show overall resource consumption during certain period
  - Find top queries by their execution time or resource consumption
  - Track specific query overtime performance

Confidential

Query Store Demo

```
-- Create table and index
CREATE TABLE Quotations
(
ID INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
Name VARCHAR(100) NOT NULL,
Quote VARCHAR(5000) NOT NULL,
Extra VARCHAR(100),
Iteration INT  NOT NULL
)
GO
CREATE NONCLUSTERED INDEX idx_Iteration ON Quotations(Iteration)

-- Create a procedure to query data based on parameter
CREATE PROCEDURE RetrieveQuotes
(
@Value INT
)
AS
BEGIN
SELECT * FROM Quotations
WHERE Iteration <  @Value
END
GO
```

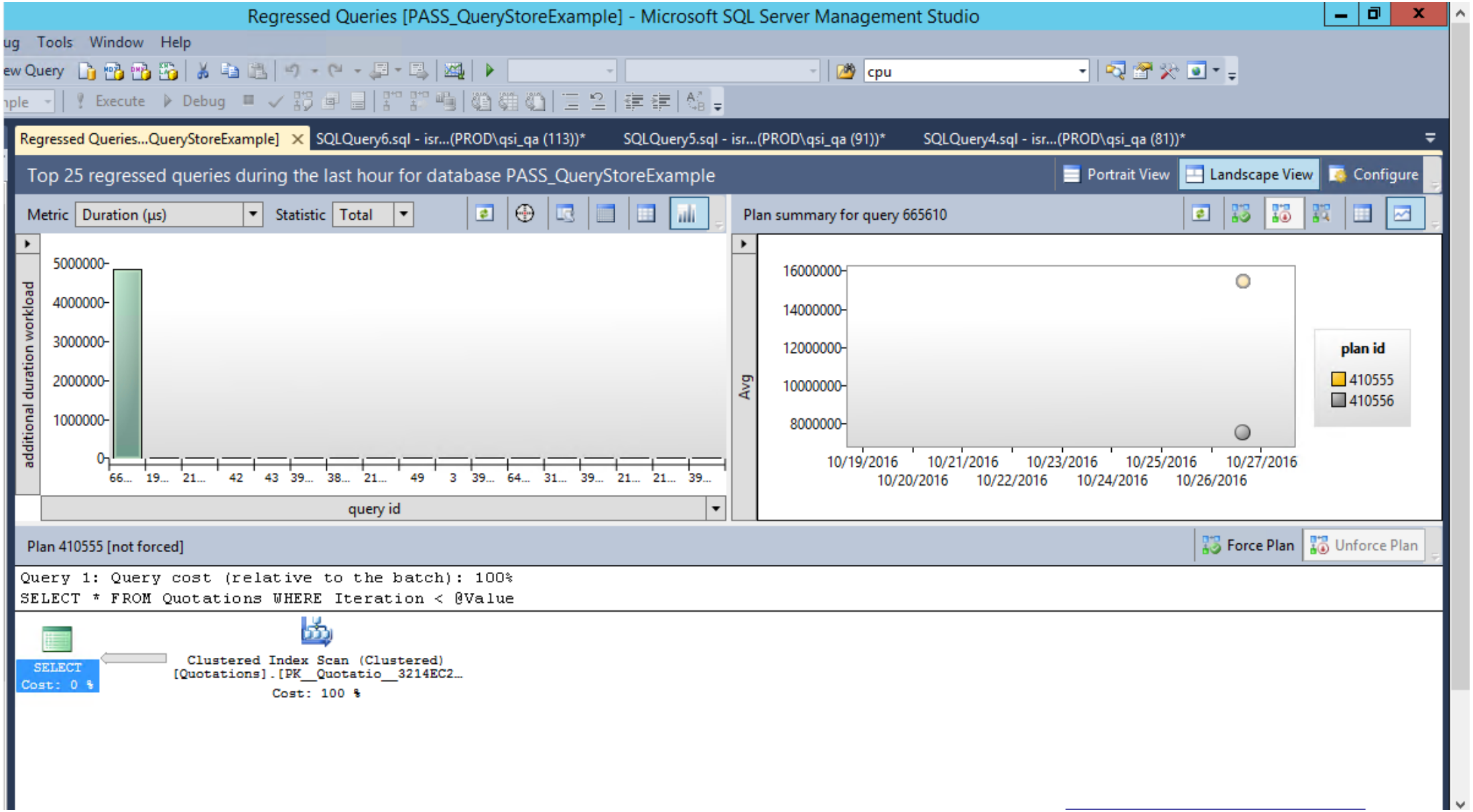

Confidential

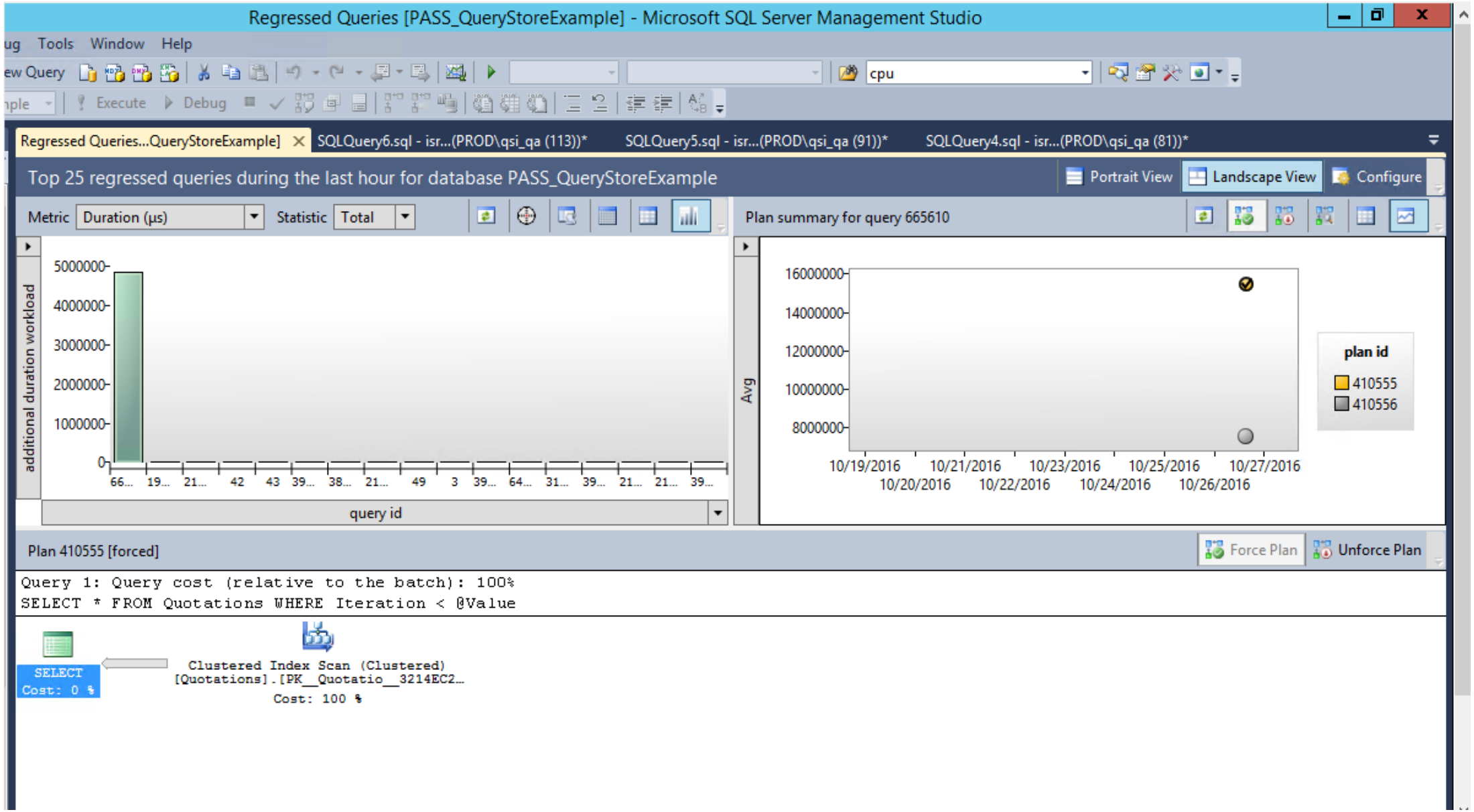
Quest™

Query Store Demo

```
-- Enable live query statistics
set statistics io on
GO
-- Query the data based on parameter that will return many rows
exec RetrieveQuotes 30000
-- now simulate restart
DBCC FREEPROCCACHE
GO
exec RetrieveQuotes 2
-- read 163 pages using index
-- now we go back to use 30000 again
exec RetrieveQuotes 30000
-- now instead of 15000 pages, there are almost 3M logical reads!
```


Confidential
Quest™


Regressed Queries [PASS_QueryStoreExample] - Microsoft SQL Server Management Studio
Tools
Window
Help
cpu
Execute
Debug
Regressed Queries...QueryStoreExample]
SQLQuery6.sql - isr...(PROD\qsi_qa (113))*
SQLQuery5.sql - isr...(PROD\qsi_qa (91))*
SQLQuery4.sql - isr...(PROD\qsi_qa (81))*
Top 25 regressed queries during the last hour for database PASS_QueryStoreExample
Portrait View
Landscape View
Configure
Metric
Duration (µs)
Statistic
Total
Plan summary for query 665610
additional duration workload
5000000
4000000
3000000
2000000
1000000
0
66...
19...
21...
42
43
39...
38...
21...
49
3
39...
64...
31...
39...
21...
21...
39...
query id
Avg
16000000
14000000
12000000
10000000
8000000
10/19/2016
10/20/2016
10/21/2016
10/22/2016
10/23/2016
10/24/2016
10/25/2016
10/26/2016
10/27/2016
plan id
410555
410556
Plan 410555 [not forced]
Force Plan
Unforce Plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Quotations WHERE Iteration < @Value
SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[Quotations].[PK__Quotatio__3214EC2...
Cost: 100 %

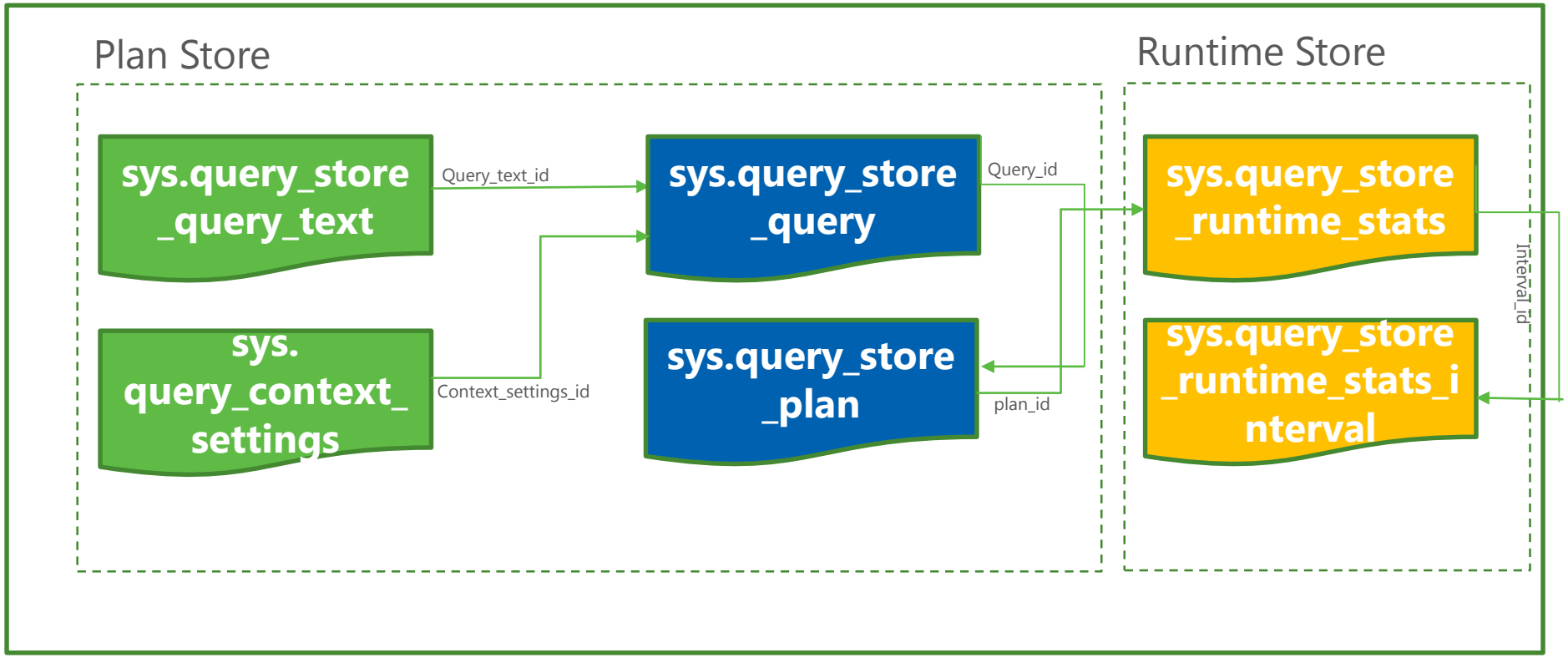

Regressed Queries [PASS_QueryStoreExample] - Microsoft SQL Server Management Studio
Tools
Window
Help
New Query
Execute
Debug
cpu
Regressed Queries...QueryStoreExample]
SQLQuery6.sql - isr...(PROD\qsi_qa (113))*
SQLQuery5.sql - isr...(PROD\qsi_qa (91))*
SQLQuery4.sql - isr...(PROD\qsi_qa (81))*
Top 25 regressed queries during the last hour for database PASS_QueryStoreExample
Portrait View
Landscape View
Configure
Metric
Duration (µs)
Statistic
Total
additional duration workload
5000000
4000000
3000000
2000000
1000000
0
66...
19...
21...
42
43
39...
38...
21...
49
3
39...
64...
31...
39...
21...
21...
39...
query id
Plan summary for query 665610
Avg
16000000
14000000
12000000
10000000
8000000
10/19/2016
10/20/2016
10/21/2016
10/22/2016
10/23/2016
10/24/2016
10/25/2016
10/26/2016
10/27/2016
plan id
410555
410556
Plan 410555 [forced]
Force Plan
Unforce Plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Quotations WHERE Iteration < @Value
SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[Quotations].[PK__Quotatio__3214EC2…
Cost: 100 %

Query Store Benefits

- Easy to use
- Highly customizable to support various use cases and workloads
- Granular customization at database level
- Easy to manage at query level
- Automatic storage management
- Rich set of statistics and easy access enables track of many types of problems
- Embedded within the database engine ensure nothing is missed. Including SQL texts!
- Support natively compiled procedures and in-memory OLTP workload

Confidential

Query Store Schema



Confidential

Quest

Query Store Schema Data

Text	Query	Plan	Runtime stats
• Text id • Query text	• Query_id • Query_hash • Compile stats • Batch_sql handle • Execution Time	• plan_id • Plan xml • Compile statistics	• Resource consumption stats: CPU, memory, reads • Executions count • Status • Duration • CLR Time • Degree of Parallelism

Type of stats: avg, min, max, std, last.
For sum : avg * executions_count

Confidential
Quest™

Query Store Additional Use Cases

Using Custom Queries

- Compare activity between different time frames
- Find queries that exceeds certain duration threshold
- Find frequently failed queries
- Identify queries that suffer plan instability
- Find top compile resources consumers
- Find queries that needs to be parameterized
- Find top regressed queries
- And more..

Confidential

Querying the Query Store

- Note the aggregation level and time of aggregation
- Statistics that are not yet aggregated may cause duplicates values

SQLQuery4.sql - isr...(PROD\qsi_qa (277))* | SQLQuery3.sql - isr...(PROD\qsi_qa (272))* | Regressed Queries...QueryStoreExample]

```
select plan_id,runtime_stats_interval_id,execution_type,max(first_execution_time),max(last_execution_time),count(*)
from sys.query_store_runtime_stats
group by plan_id,runtime_stats_interval_id,execution_type
having count(*) > 1
```

100 %

Results | Messages

	plan_id	runtime_stats_interval_id	execution_type	(No column name)	(No column name)	(No column name)
1	151	102	0	2016-10-22 15:01:09.1600000 +00:00	2016-10-22 15:28:46.8600000 +00:00	15
2	238	102	0	2016-10-22 15:02:25.8770000 +00:00	2016-10-22 15:28:44.4970000 +00:00	1290
3	2088	102	0	2016-10-22 15:01:40.6730000 +00:00	2016-10-22 15:22:18.3900000 +00:00	2
4	322946	102	0	2016-10-22 15:03:53.5770000 +00:00	2016-10-22 15:20:44.9300000 +00:00	2
5	1522	102	0	2016-10-22 15:03:44.8030000 +00:00	2016-10-22 15:26:43.4230000 +00:00	7
6	3478	102	0	2016-10-22 15:03:53.5270000 +00:00	2016-10-22 15:25:58.8500000 +00:00	3
7	526	102	0	2016-10-22 15:01:13.3330000 +00:00	2016-10-22 15:28:12.8330000 +00:00	7
8	952	102	0	2016-10-22 15:01:26.1130000 +00:00	2016-10-22 15:28:18.1970000 +00:00	7
9	429	102	0	2016-10-22 15:01:14.9370000 +00:00	2016-10-22 15:28:27.0000000 +00:00	11
10	2366	102	0	2016-10-22 15:01:20.5700000 +00:00	2016-10-22 15:24:13.3870000 +00:00	8

Query executed successfully. | isrvmn805\x64_2016_yr (13.0... | PROD\qsi_qa (277) | PASS_QueryStoreExample | 00:00:34 | 757 rows

Confidential

Quest

The Bad
Limitations of Query Store

Quest

Query Store Enabled at Database Level

- Cannot be enabled for master and tempdb
- Does not work on Read Only databases
  - No read only AG replicas
- No way to set an instance default
  - Must be enabled for new instances
- Multiple DBA’s could change capture mode, hard to track
- No way to globally control configuration
  - Without manual scripting

Confidential

Space Consumption Must be Managed

- When Query Store reaches capacity, turns read only
  - May not have it when you need it most
- Balancing act between space consumed and history required
- Data is stored to primary filegroup
  - IO contention with user data
  - Longer database restores

Confidential

How Much Disk Space is Required

- **Configuration Parameters**
  - INTERVAL_LENGTH_MINUTES (60) - 1, 5, 10, 15, 30, 60, 1440
  - MAX_STORAGE_SIZE_MB (100) - Maximum size of the query store
  - QUERY_CAPTURE_MODE (ALL) - ALL/AUTO/NONE
  - MAX_PLANS_PER_QUERY (200) - 0 is no limit
  - CLEANUP_POLICY (367) - 0 to disable
  - SIZE_BASED_CLEANUP_POLICY - AUTO or OFF

- **Database Workload**
  - Number of different query texts
  - Using non-parameterized queries can cause excessive “similar” queries

Confidential

Little Context: How Was Query Run?

- Query Store provides detailed metrics about query performance
- Little information is provided about context
  - Who ran a particular query
  - Which programs and/or servers did the queries execute from
- Useful for general performance tuning, not for finger pointing

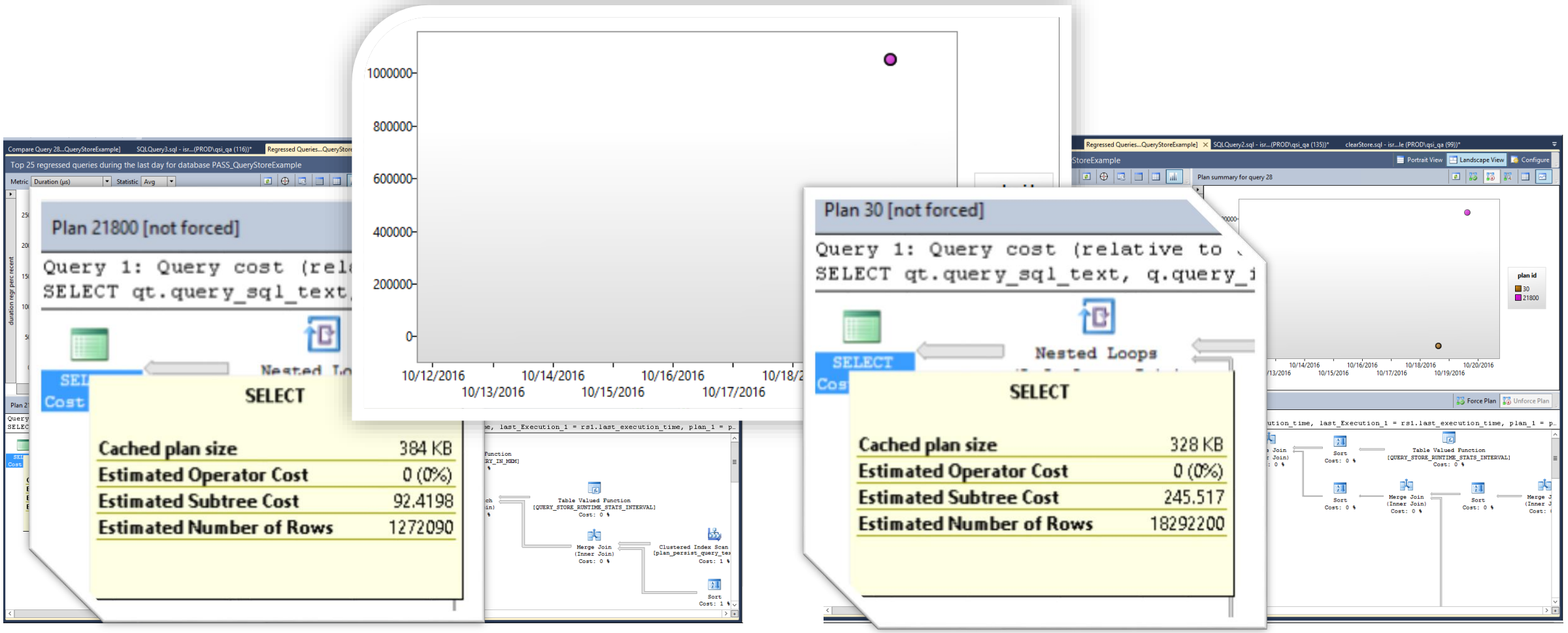
Confidential

Execution Plan “Pinning”, Good or Bad

- “Pinning” an execution plan may provide a quick fix
  - Fix may be temporary
- What if the schema changed?
- What if your query changes slightly?
- What if the data volumes change?
- What if significant time has passed?
- Worry about overly “pinned” systems degrading over time

Confidential

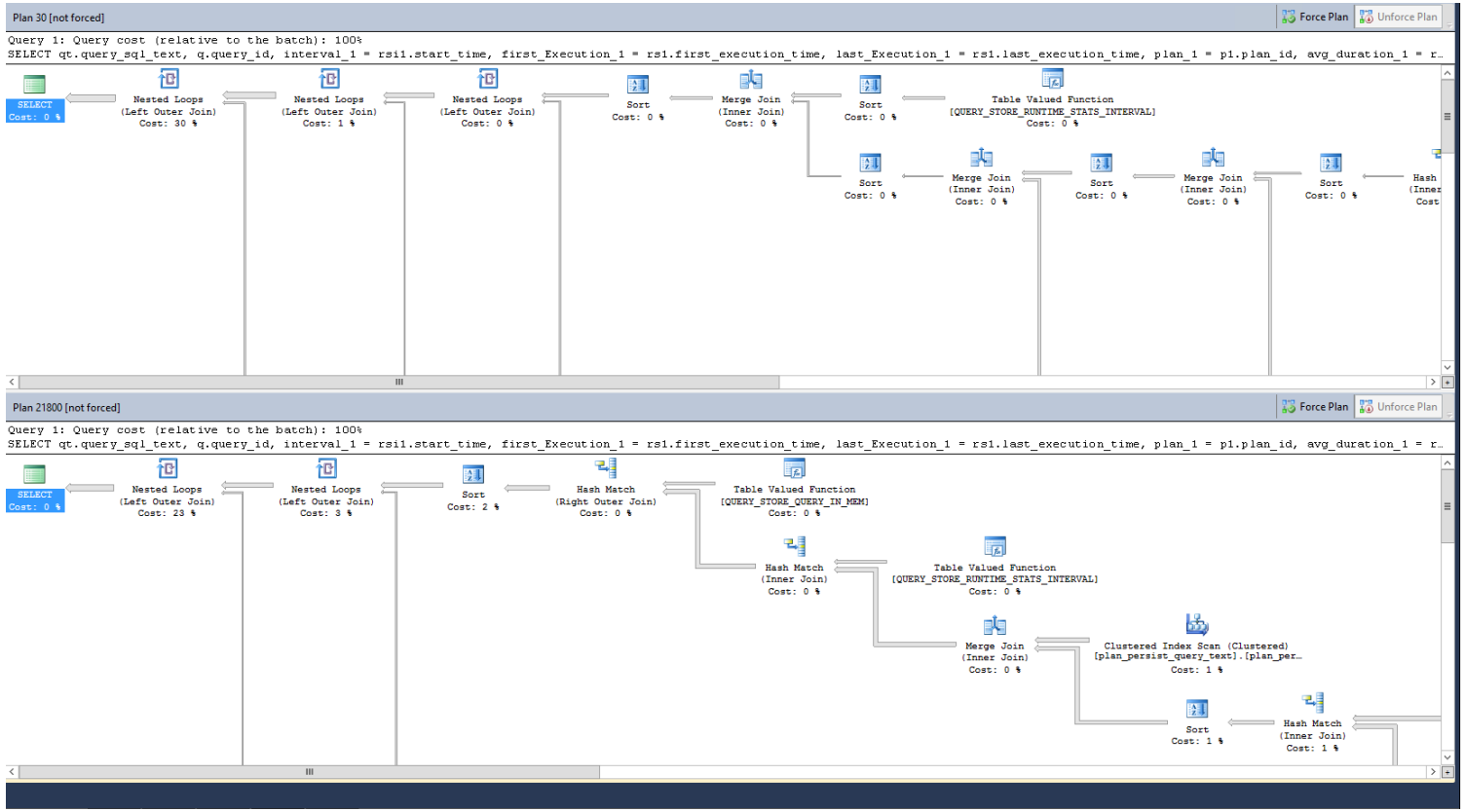
Plan Forcing – A Good Thing?



Confidential

Quest

Plan Compare



Confidential

Quest

Overhead?

- Configuration Parameters
  - Length of interval controls frequency of aggregations and amount of rows in table
  - Frequency of cleans
  - Capture mode control amount of queries to capture
  - Flush interval control amount of memory used
- Database Workload
  - Amount of different query texts and use of parameterization
- Query Store usage
  - Amount of queries run against the store and their type

Confidential

Monitor – Perfmon Counters

Host and DB

```
Memory\% Committed Bytes In Use
MSSQL$X64_2016_YR:Buffer Manager\Page life expectancy
MSSQL$X64_2016_YR:Buffer Manager\Page lookups/sec
MSSQL$X64_2016_YR:Buffer Manager\Page reads/sec
MSSQL$X64_2016_YR:Buffer Manager\Page writes/sec
MSSQL$X64_2016_YR:Transactions\Transactions
PhysicalDisk(_Total)\Disk Transfers/sec
Processor(_Total)\% Processor Time
```

Query Store

```
:Query Store(_Total)\Query Store CPU usage
:Query Store(_Total)\Query Store logical reads
:Query Store(_Total)\Query Store logical writes
:Query Store(_Total)\Query Store physical reads
```


Confidential

36

Informal Performance Benchmark

- Use Benchmark factory to run TPC-E like workload.
- 150 Users. Database scale 1 (around 10GB)
- Same machine – windows 2012, 4 CPUs , 8GB RAM, VM
- Duration – 30 minutes
- With Query Store and without
- Capture mode is ALL
- Use perfmon data set collector to monitor
- SQL Server version is CTP3.2

Confidential

Performance Benchmark Summary

Type	%ProcessorTime	Query Store CPU Usage	Disk Transfer/s	Query Store Physical Reads	%Comitted Bytes in Use	Page Writes/s	Query Store logical Writes
Without QS	64.22%	-	489.811	-	81%	128.6 (max 737)	-
With QS	81.2%	20.752	683.478	13.705 (max 211)	85%	142.7 (max 629)	34.6

Confidential

Monitor Waits and Extended Events

Extended Events

name	description
query_store_failed_to_capture_query	Fired if the Query Store failed to capture query. The Query Store will not track statistics for this query
query_store_failed_to_load_forced_plan	Fired if the query failed to load forced plan from Query Store. Forcing policy will not be applied
query_store_failed_to_find_resource_group	Fired when Query Store resource group is not initialized
query_store_persist_on_shutdown_failed	Occurs when SQL Server fails to store dirty entries in Query Store on database shutdown.
query_store_begin_persist_runtime_stat	Fired immediately before current runtime statistics for a query plan is persisted to the database.
query_store_execution_runtime_info	Fired when runtime information is sent to the Query Store.
query_store_execution_runtime_info_discarded	Fired when runtime information sent to the Query Store is discarded.
query_store_execution_runtime_info_evicted	Fired when runtime information sent to the Query Store is evicted.
query_store_statement_not_found	Fired in case when statement couldn't be found due to race condition or ambiguous user request.
query_store_plan_forcing_failed	Occurs when forcing of plan from Query Store fail
query_store_background_task_creation_failed	Fired if the background processing task for Query Store could not be created
query_store_background_task_initialization_failed	Fired if the background processing task for Query Store could not be initialized
query_store_background_task_persist_started	Fired if the background task for Query Store data persistence started execution
query_store_background_task_persist_finished	Fired if the background task for Query Store data persistence is completed successfully
query_store_background_task_persist_failed	Fired if the background task for Query Store data persistence is not completed successfully
query_store_disk_size_info	Fired when a check against Query Store on-disk size is performed
query_store_disk_size_check_failed	Fired when a check against Query Store on-disk size limit fails
query_store_stmt_hash_map_over_memory_limit	Fired when Query Store statement hash map memory size grows over allowed memory limit

Waits

wait_type	wait_time_ms
QDS_DYN_VECTOR	0
QDS_STMT	0
QDS_CTXS	0
QDS_BCKG_TASK	0
QDS_DB_DISK	0
QDS_STMT_DISK	0
QDS_ASYNC_PERSIST_TASK	0
QDS_LOADDB	0
QDS_ASYNC_PERSIST_TASK_START	0
QDS_ASYNC_CHECK_CONSISTENCY_TASK	0
QDS_TASK_START	3
QDS_PERSIST_TASK_MAIN_LOOP_SLEEP	621810742
QDS_TASK_SHUTDOWN	0
QDS_SHUTDOWN_QUEUE	0
QDS_EXCLUSIVE_ACCESS	0
QDS_CLEANUP_STALE_QUERIES_TASK_MAIN_LOOP_SLEEP	0
QDS_ASYNC_QUEUE	0
QDS_BLOOM_FILTER	0
QDS_QDS_CAPTURE_INIT	0

Confidential

Other Limitations

- Queries with multiple literal values, multiple entries
- Queries that access multiple databases
- Linked server queries, only logged in source
- No information provided related to waits
- One time interval – difficult to see information for large time frames
- Cumbersome to write queries against the store

Confidential

Be A Hero
The Query Store Sweet Spot

Quest

Know the Sweet Spot

- Before rolling a major change to production
  - Ensure no errors
  - Tune high resource usage queries
- Before and After a major change to hardware or software versions
  - To compare performance trend and resource usage
  - To ensure no query has regressed
- Keep performance stability during the upgrade to SQL Server 2016
  - Upgrade to 2016 but set compatibility level to version before upgrade
  - Enable query store to capture baseline
  - Change compatibility level to 130
  - Examine changes and find regressed queries

Confidential

Share Results Gathered by Query Store

```
-- Compare top statements by duration over period of 2 days
Declare @history_end_time datetimeoffset(7),
@history_start_time datetimeoffset(7),
@results_row_count int = 1000;

set @history_end_time = GETUTCDATE();
set @history_start_time = DATEADD(day, -2, GETUTCDATE());

with stats as(
    SELECT         p.query_id query_id,
                rs.runtime_stats_interval_id,
                CONVERT(float, SUM(rs.avg_cpu_time*rs.count_executions)) total_cpu_time,
                 CONVERT(float, SUM(rs.avg_duration*rs.count_executions)) total_duration,
                 CONVERT(float, SUM(rs.avg_logical_io_writes*rs.count_executions)) total_logical_io_writes,
                 CONVERT(float, SUM(rs.avg_logical_io_reads*rs.count_executions)) total_logical_io_reads,
                 CONVERT(float, SUM(rs.avg_query_max_used_memory*rs.count_executions)) total_query_max_used_memory,
                 CONVERT(float, SUM(rs.avg_physical_io_reads*rs.count_executions)) total_physical_io_reads,
                 SUM(rs.count_executions) count_executions,
                AVG(rs.avg_duration) avg_duration,
                row_number() over (partition by runtime_stats_interval_id order by SUM(rs.avg_duration*rs.count_executions) desc) as row#
    FROM sys.query_store_runtime_stats rs
        JOIN sys.query_store_plan p ON p.plan_id = rs.plan_id
    WHERE NOT (rs.first_execution_time > @history_end_time OR rs.last_execution_time < @history_start_time) and execution_type = 0
    GROUP BY rs.runtime_stats_interval_id,p.query_id
)
select *
from stats
where row# <= 10
```


Confidential

Quest

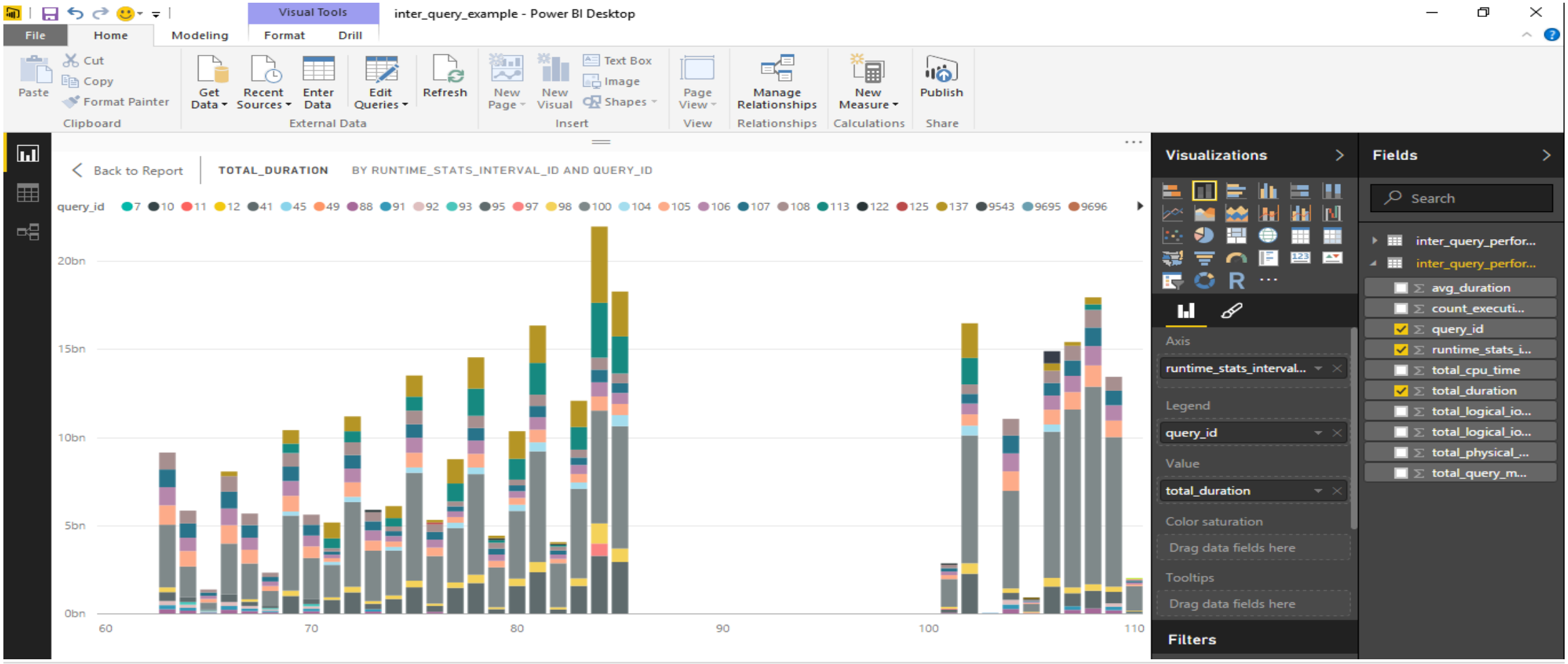
Results | Messages

	query_id	runtime_stats_interval_id	total_cpu_time	total_duration	total_logical_io_writes	total_logical_io_reads	total_query_max_used_memory	total_physical_io_reads	count_executions	avg_duration	row#
1	7	319	115019	141012	7	3919	0	0	86	1639.67441860465	1
2	10	319	6936	6936	0	189	0	0	7	990.857142857143	2
3	7	320	109958	148964	4	3660	0	0	83	1794.74698795181	1
4	665659	320	15982	94005	0	966	792	97	3	31335	2
5	10	320	7999	7999	0	189	0	0	7	1142.71428571429	3
6	12	320	999	999	0	44	0	0	1	999	4
7	11	320	0	0	0	0	0	0	1	0	5
8	7	321	116955	147946	4	3748	0	0	85	1740.54117647059	1
9	665659	321	13999	22003	0	849	792	36	3	7334.33333333333	2
10	10	321	7996	9997	0	189	0	0	7	1428.14285714286	3
11	12	321	1999	1999	0	44	0	0	1	1999	4
12	11	321	0	0	0	0	0	0	1	0	5
13	7	322	113905	140903	5	3840	0	0	87	1619.57471264368	1
14	665659	322	20013	100007	0	849	792	34	3	33335.6666666667	2
15	10	322	7989	7989	0	189	0	0	7	1141.28571428571	3
16	12	322	3999	3999	0	132	0	0	3	1333	4
17	11	322	0	0	0	0	0	0	3	0	5
18	665659	323	22995	504026	0	849	792	95	3	168008.666666667	1
19	7	323	112947	123948	1	3786	0	0	86	1441.25581395349	2
20	10	323	6985	7984	0	189	0	0	7	1140.57142857143	3
21	12	323	4000	4000	0	88	0	0	2	2000	4
22	11	323	1000	1000	0	0	0	0	2	500	5
23	7	324	124982	141984	5	3867	0	0	85	1670.4	1
24	665659	324	17999	35998	0	849	792	18	3	11999.3333333333	2
25	10	324	5986	10987	1	191	0	0	7	1569.57142857143	3
26	12	324	1000	1000	0	44	0	0	1	1000	4
27	11	324	0	0	0	0	0	0	1	0	5
28	665659	325	22988	274016	0	849	792	36	3	91338.6666666667	1
29	7	325	127952	146956	14	3761	0	0	85	1728.89411764706	2
30	10	325	8990	8990	0	189	0	0	7	1284.28571428571	3
31	12	325	1999	1999	0	44	0	0	1	1999	4
32	11	325	0	0	0	0	0	0	1	0	5
33	665659	326	16000	180018	0	849	792	36	3	60006	1
34	7	326	127980	145982	7	3838	0	0	87	1677.95402298851	2
35	39	326	0	23996	0	20	128	2	1	23996	3
36	10	326	7995	8996	0	189	0	0	7	1285.14285714286	4
37	38	326	4999	4999	0	35	128	31	1	4999	5
38	12	326	3993	3993	1	134	0	0	3	1331	6
39	11	326	0	0	0	0	0	0	3	0	7

Confidential

Quest

View Results with Power BI



Confidential

Quest

Best Practices

Quest

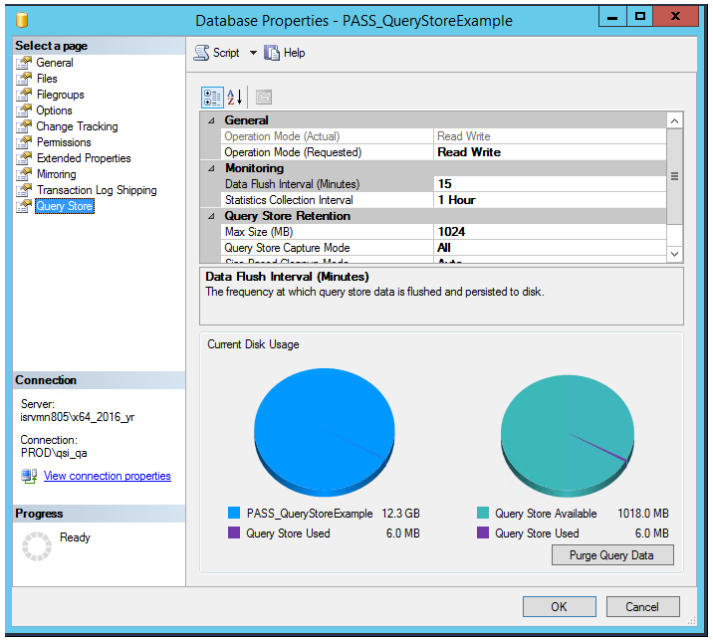
Monitor Query Store Size

T-SQL

```
SELECT
current_storage_size_mb,
max_storage_size_mb ,
current_storage_size_mb*100
/max_storage_size_mb as
query_store_utilization

FROM
sys.database_query_store_op
tions;
```

SSMS



Confidential

Quest

Monitor Query Store Collection State

```
Select desired_state_desc,actual_state_desc,readonly_reason
FROM sys.database_query_store_options;
```

1 – database is in read-only mode

2 – database is in single-user mode

4 – database is in emergency mode

8 – database is secondary replica

65536 – the Query Store has reached the size limit set by the MAX_STORAGE_SIZE_MB option.

Confidential

Quest™

Pay Attention to Capture Mode

- CAPTURE_MODE control the type and amount of queries Query Store will monitor
  - All
  - Auto – only capture information about top queries in term of resource usage. Ignore infrequent queries
  - None – will not gather information about new queries.
- Change Capture Mode according to scenario
- Can monitor via the sys.database_query_store_options

Confidential

Avoid Using Non-Parameterized Queries

- Non parameterized queries waste compile resources and cause more resources to be consumed by query store
- Use query provided above to track such queries
- Possible fixes
  - Replace ad-hoc queries with stored procedure if possible
  - Use forced parameterization

Confidential

Verify Status of Forced Plans

```
select plan_id,
   query_id,
   force_failure_count,
   last_force_failure_reason_desc
from sys.query_store_plan
where is_forced_plan > 0 and force_failure_count > 0
```


Confidential
Quest™

Avoid DROP and CREATE for Database Objects

- Query id is calculated based on text and containing object id
- When containing object is recreated, a new query will be generated for the same text
- Limits the ability to track query performance over time
- Use more space
- Use ALTER instead

Confidential

Summary

- Query Store is a great resource for performance tuning
- Use it with care
- Share the use cases!

Confidential

References

- https://msdn.microsoft.com/en-us/library/dn817826.aspx
- https://msdn.microsoft.com/en-us/library/mt668803.aspx
- http://www.sqlpassion.at/archive/2016/01/18/performance-troubleshooting-with-the-query-store-in-sql-server-2016/
- https://www.simple-talk.com/sql/database-administration/the-sql-server-2016-query-store-analyzing-query-store-performance/

Confidential

Thank You
Q&A???

Quest

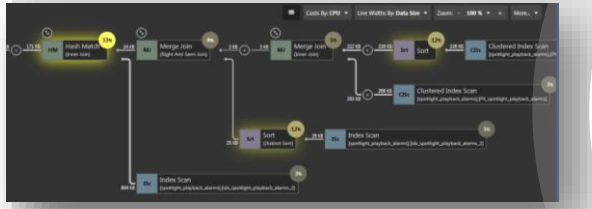
Database Performance Management Tools

Quest

Database Performance Management

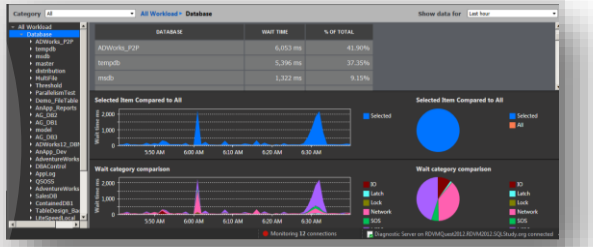
Any Time …… Any where

SQL Tuning



Faster Code

Workload Tuning



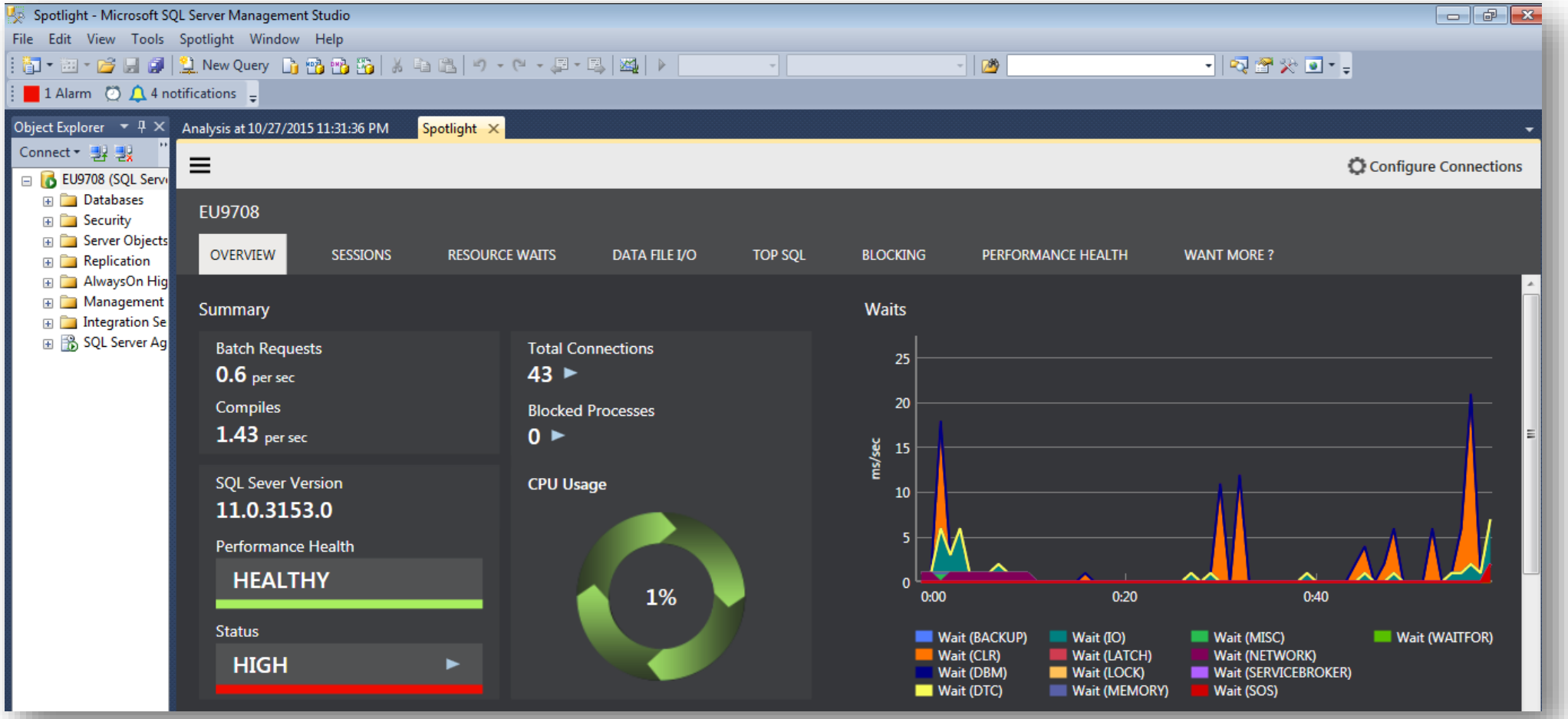
Optimized workload

On Premise … Cross Platform … Hybrid … Cloud

Confidential

Quest

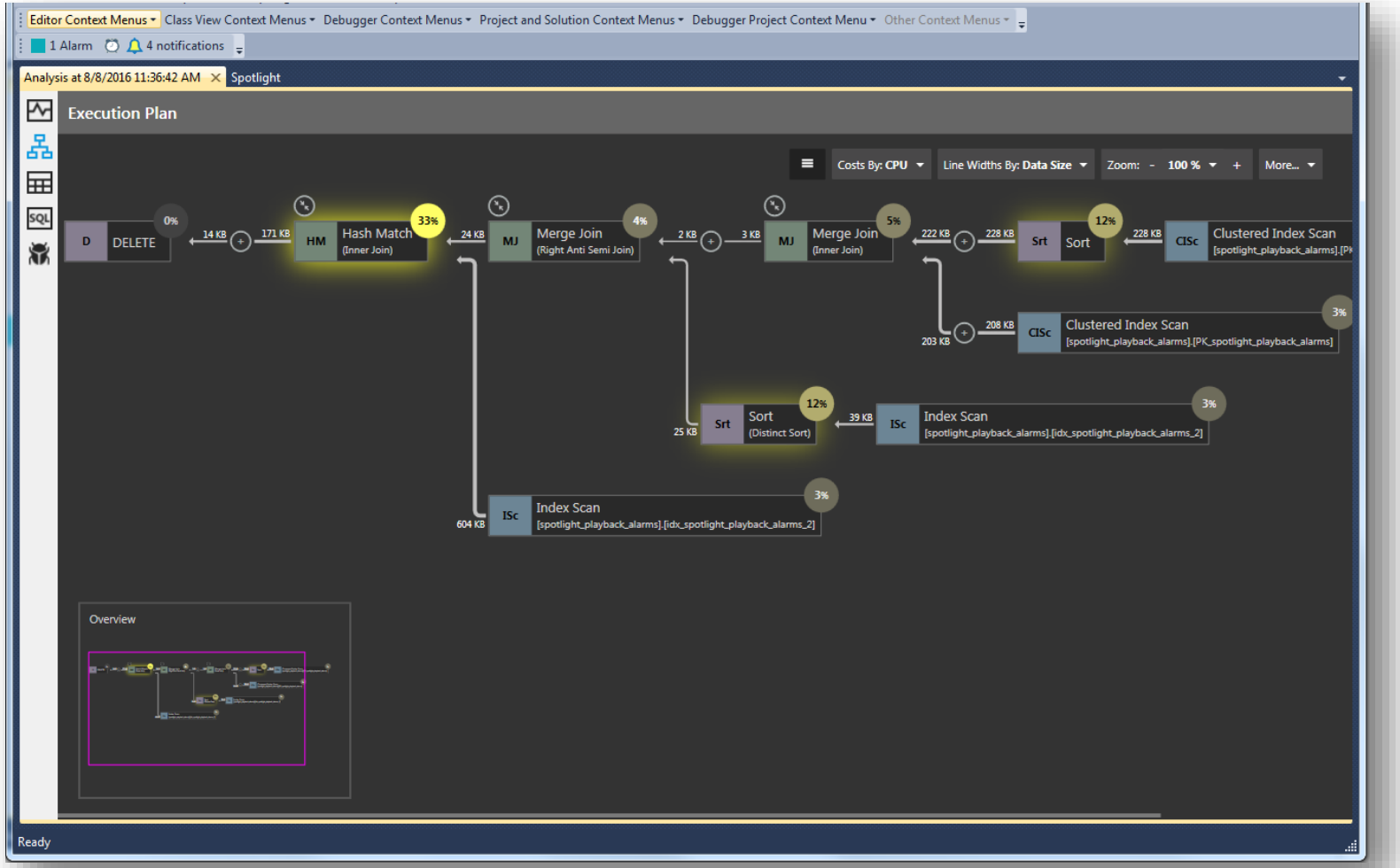
Free Diagnostics



Confidential

Quest

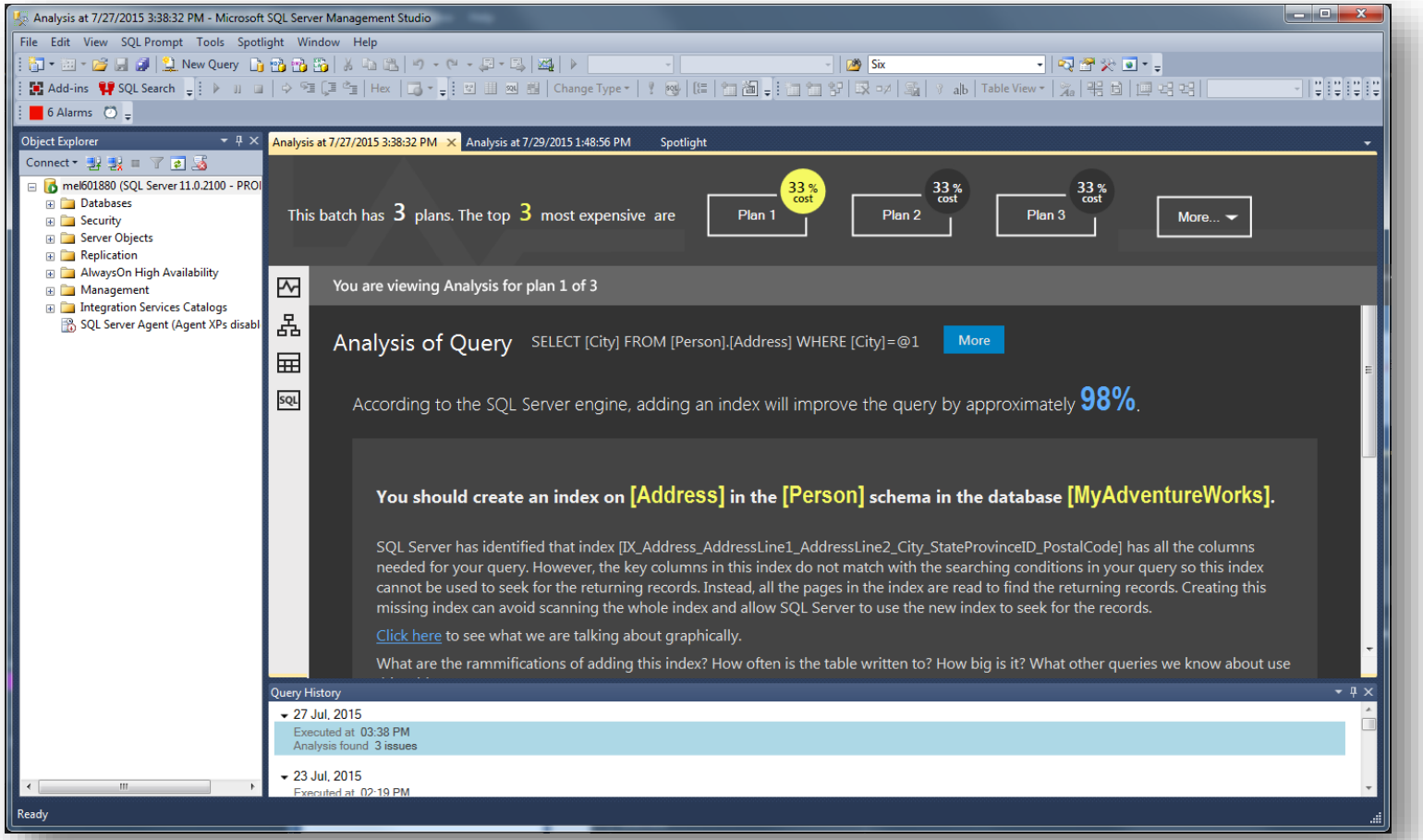
Free Plan Viewer



Confidential

Quest

Free Plan Analysis



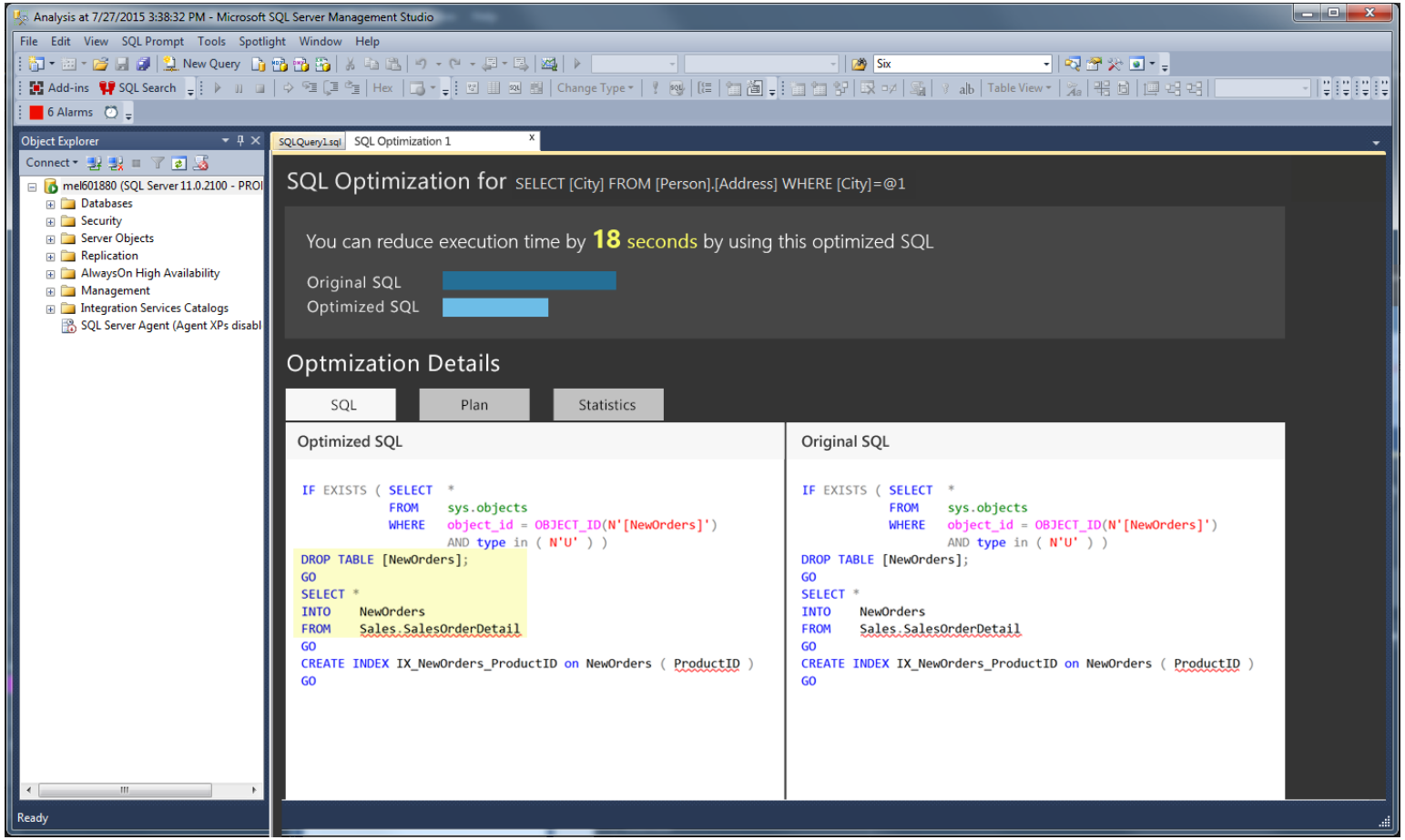
Confidential

Quest

Sign up Now at : SpotlightEssentials.com

Confidential

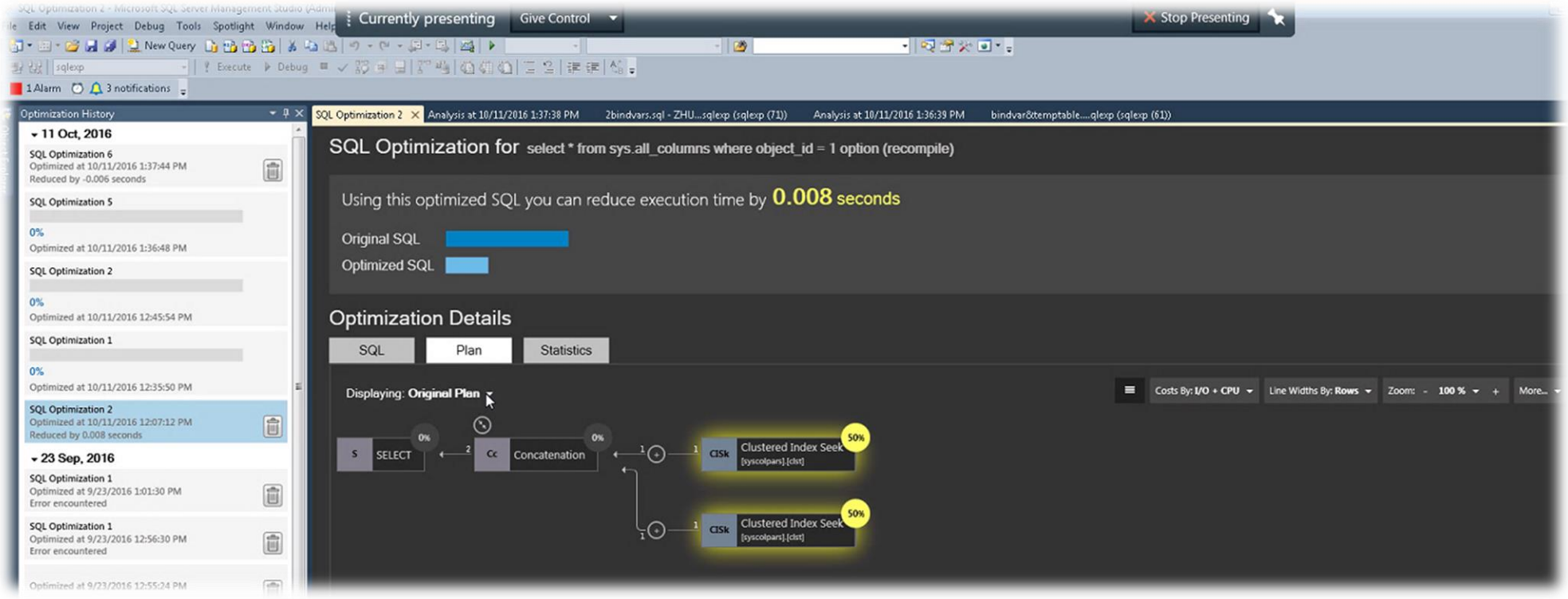
SQL Optimization



Confidential

Quest

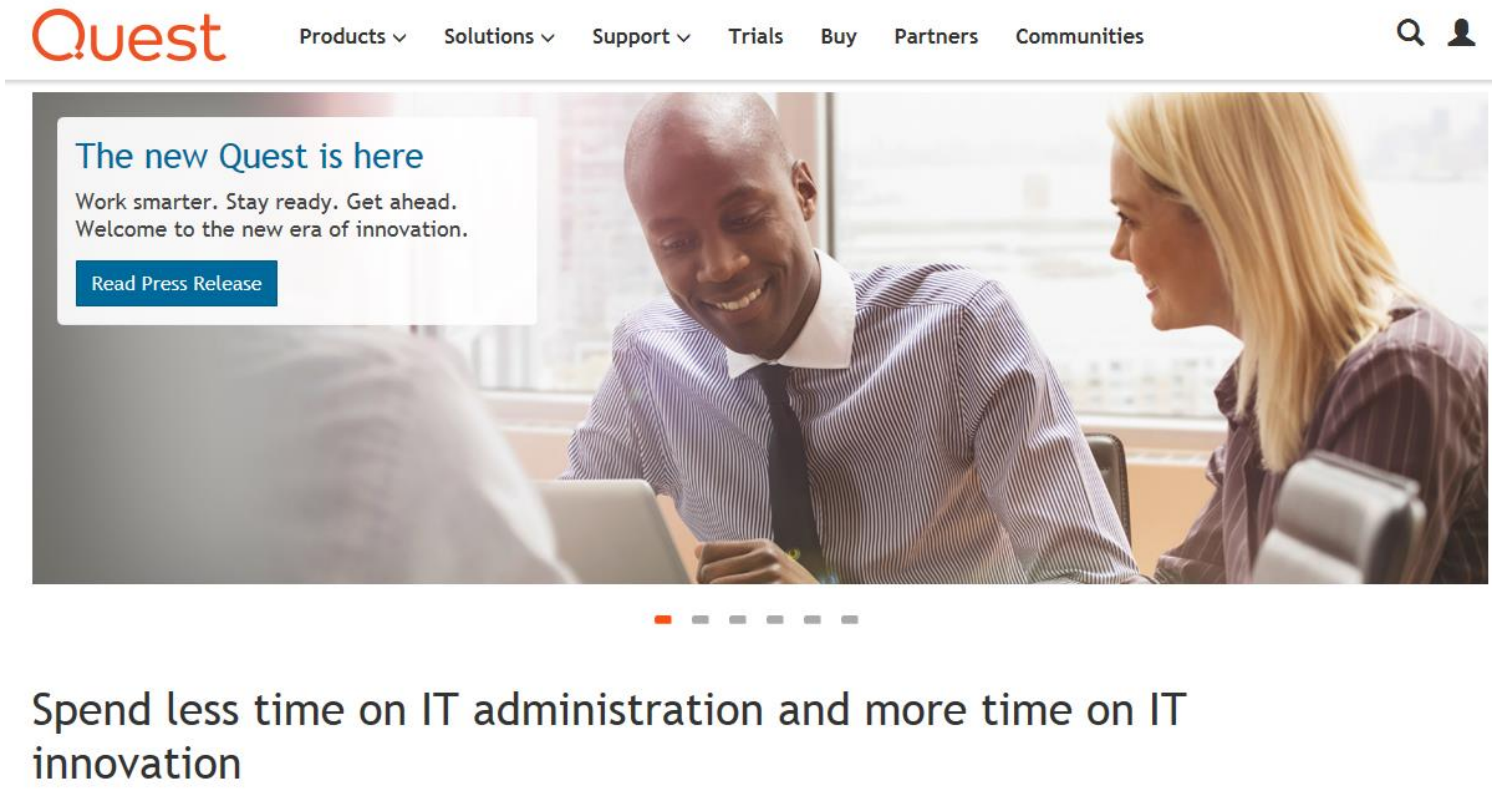
SQL Optimization



Confidential

Quest

Full range of Tools on 30 day trial at : Quest.com



Confidential