Anat Dror

PASS SQLSATURDAY ISRAEL | 11 APR 2019

What's new in Query Store 2017

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.

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

Agenda

- Introduction
    - Pre SQL Server 2016
    - Overview of Query Store
- Benefits and Gotchas
- New in SQL Server 2017
- Create a monitoring tool using the Azure Data Studio
- Best practices for managing and configuration

Overview

Pre SQL Server 2016

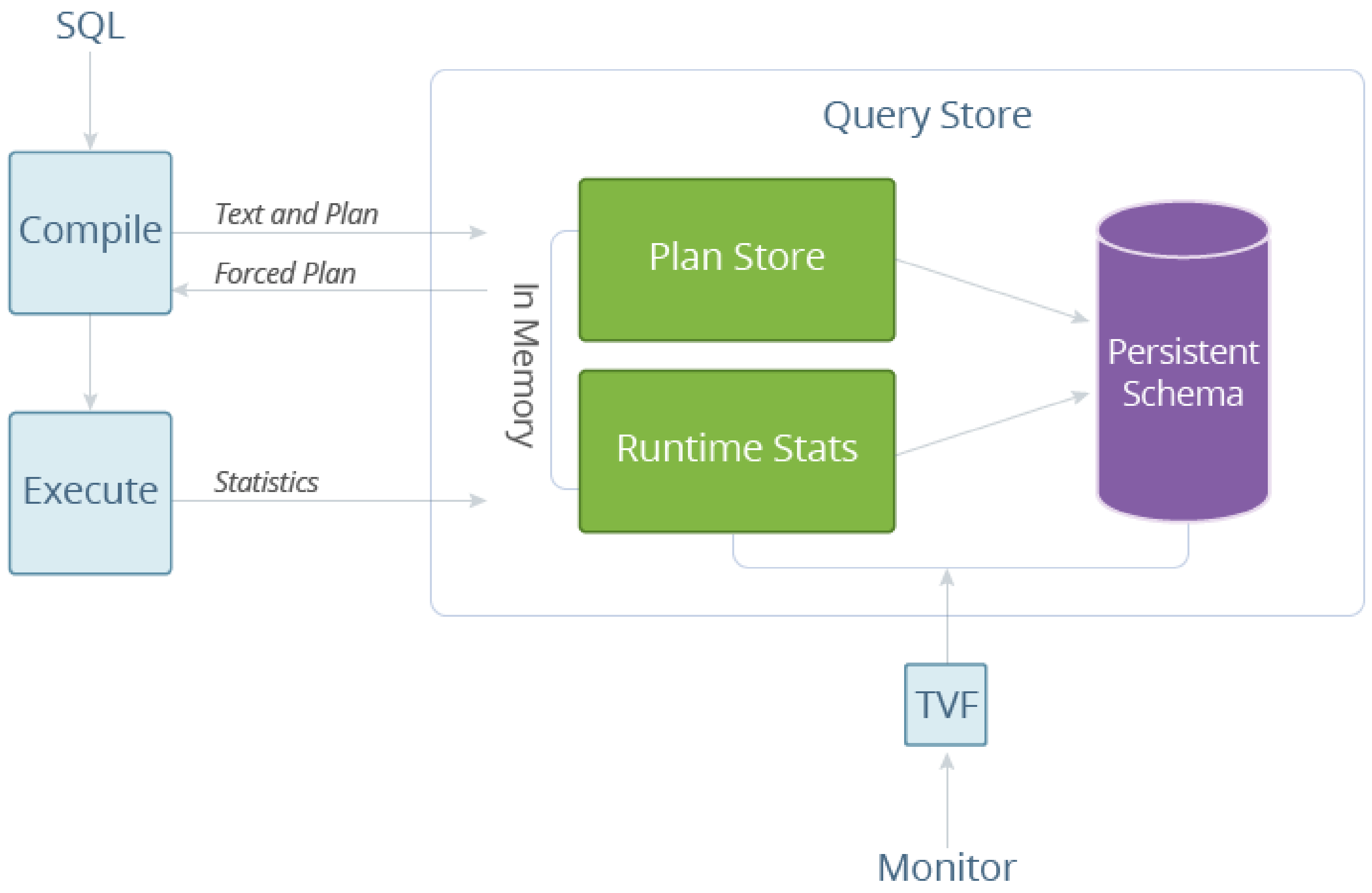
How do we understand query performance?

- 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 ...

Introducing Query Store

- New from SQL Server 2016
- Automatically captures a history of queries, plans, and runtime statistics
- Useful for troubleshooting performance degradation
- Retain over time on disk
- Aggregated by regular time windows
- Enabled at database level – you control when and where
- Can be used to track overall database workload
- Supported on all SQL Server editions

Query Store Architecture



https://docs.microsoft.com/en-us/sql/relational-databases/performance/how-query-store-collects-data?view=sql-server-2017

Query Store vs. Query Stats

Query Store	Query Stats
• Statement level	• Batch level
• Query text given	• Query text derived from batch text
• Retain over time and restarts	• Prune to memory shortage and restarts
• Aggregate by time windows	• No time window. Data is aggregated per batch, query and plan since it is logged and until it is evicted.
• Supports in-memory OLTP workloads	• No statistics about in-memory OLTP
• Include details about wait – new in 2017	• Limited wait details

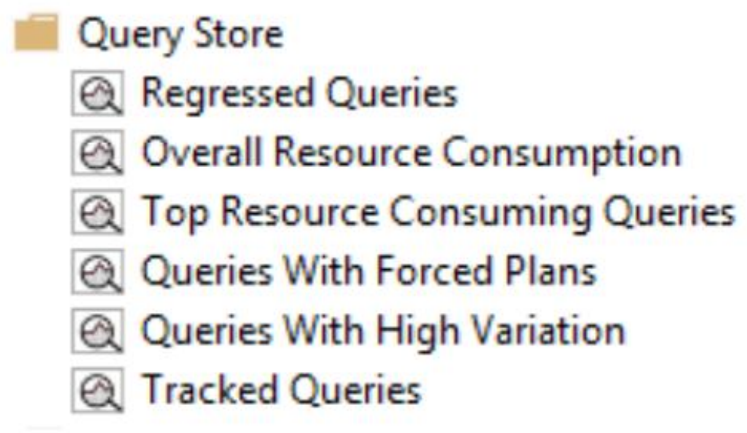
Benefits and Gotchas

Query Store Benefits

- View performance metrics broken down by time at query, plan and database level
- Audit history of query plans. Capture ALL changes to query plans.
- Rich set of statistics and easy access enables track of many types of problems
- Tracks both run time and compile time metrics
- Automatic storage management
- Embedded within the database engine ensure nothing is missed. Including SQL texts!
- Support natively compiled procedures and in-memory OLTP workload

Query Store Benefits – Cont.

- Graphical interface in SSMS to:
  - Compare plans
  - Force / Un-Force plans
  - Built-in reports/views
  - use latest SSMS version



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

- Can use T-SQL for monitoring performance
- Many SPs, QSD Wait types, & Extended Events
  - For management purposes

Use case examples

- Compare activity between different time frames
- Find queries that exceeds certain duration threshold
- Find frequently failed queries
- Find top compile resources consumers
- Find queries that needs to be parameterized
- Find top regressed queries
- Compare plans between versions of SQL Server
- And more..

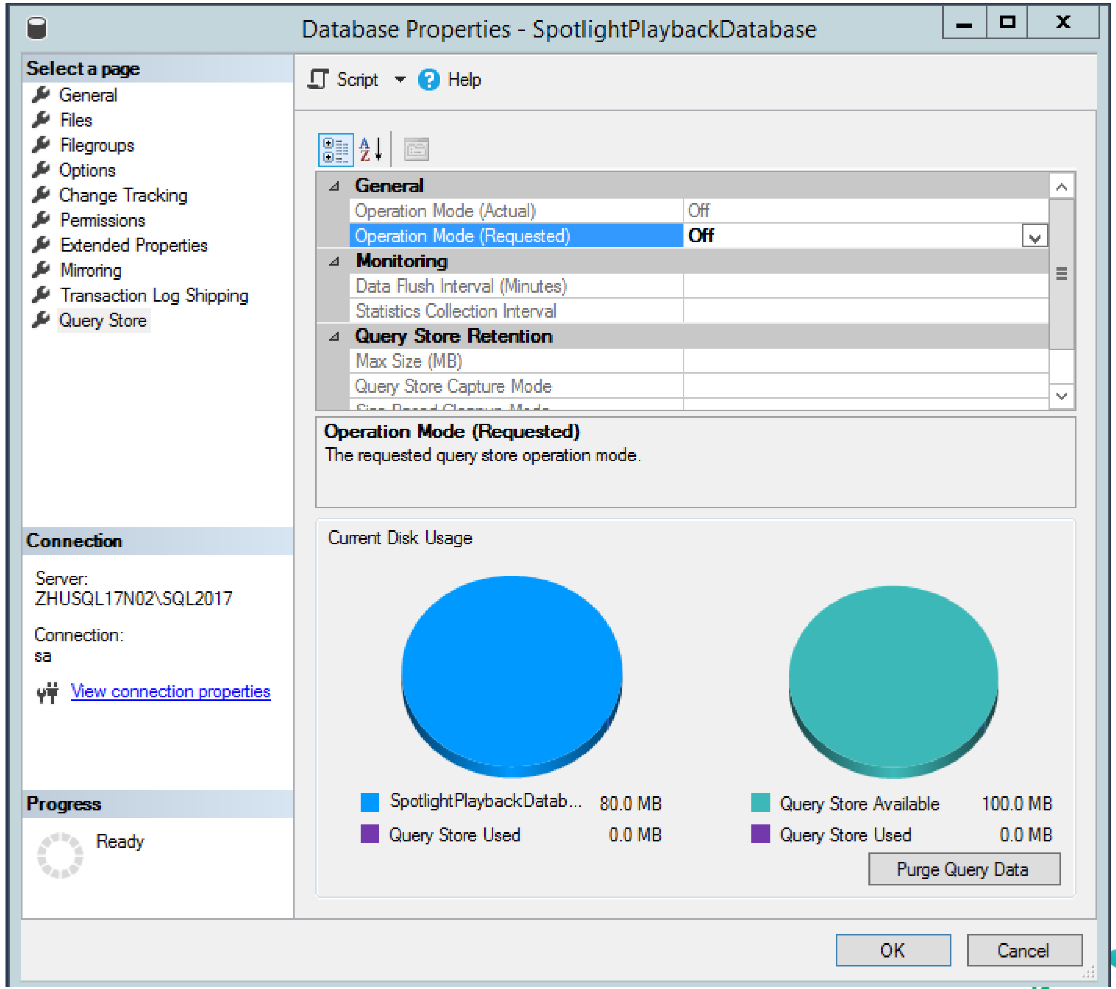
DevOps use case

- 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
    - Compare performance trend and resource usage
    - Ensure no query has regressed
- Keep performance stability during the upgrade to new SQL Server version
    - Upgrade but set compatibility level to version before upgrade
    - Enable query store to capture baseline
    - Change compatibility level to latest
    - Examine changes and find regressed queries
    - Force plan if needed

Enabling Query Store

-- T-SQL

```
ALTER DATABASE PASS_QueryStoreExample
SET QUERY_STORE = ON
GO
```



Database Properties - SpotlightPlaybackDatabase

Select a page: General, Files, Filegroups, Options, Change Tracking, Permissions, Extended Properties, Mirroring, Transaction Log Shipping, Query Store

General	
Operation Mode (Actual)	Off
Operation Mode (Requested)	Off
Monitoring	
Data Flush Interval (Minutes)	
Statistics Collection Interval	
Query Store Retention	
Max Size (MB)	
Query Store Capture Mode	

Operation Mode (Requested)
The requested query store operation mode.

Current Disk Usage

SpotlightPlaybackDatab...	80.0 MB
Query Store Used	0.0 MB
Query Store Available	100.0 MB
Query Store Used	0.0 MB

Connection

Server:
ZHUSQL17N02\SQL2017

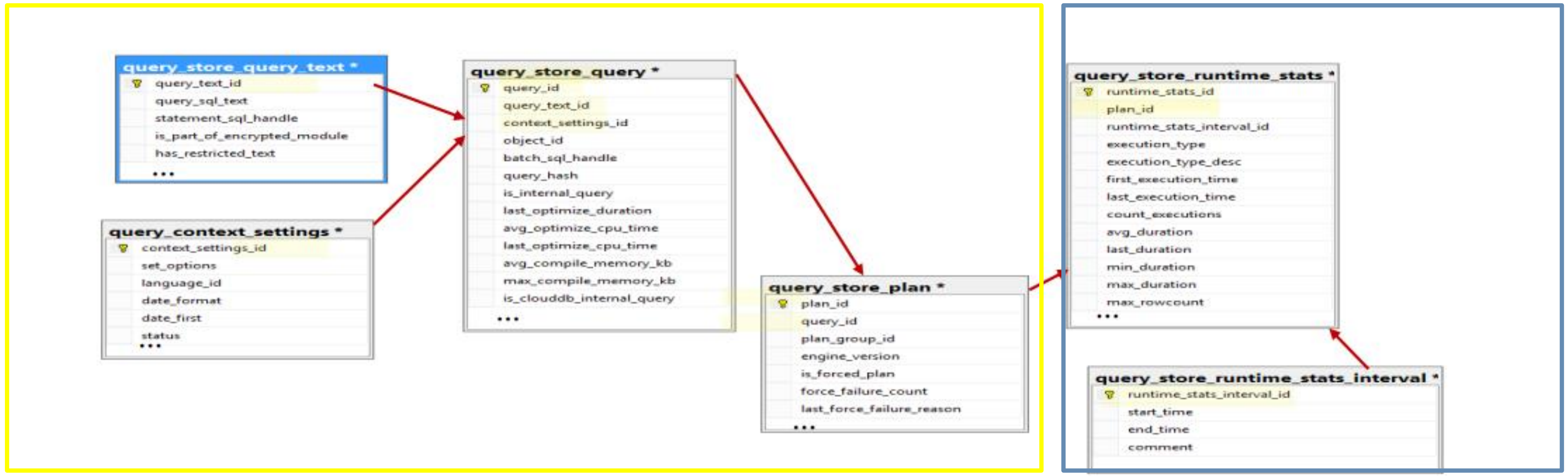
Connection:
sa

View connection properties

Progress

Ready

Query Store Schema



- query_store_query_text - text entered by user, includes whites space, hints, etc...
- query_context_settings_ - Presents unique combinations of plan affecting settings under which queries are executed
- Query_store_query – contain one row for each query id and related aggregated statistics
- Query_store_plan – contain one row for each estimated query plan created. Includes statistics about the plan and general information like forcing, if it is natively compilation etc.
- query_store_runtime_stats- runtime execution statistics for queries (avg,min,max,std deviation)
- query_store_runtime_stats_interval - start and end times intervals for statistics collected

Query Store Schema data

Text

- Text id
- Query text

Query

- Query_id
- Query_hash
- Compile stats
- Batch_sql handle
- Execution Time

Plan

- plan_id
- Plan xml
- Compile statistics

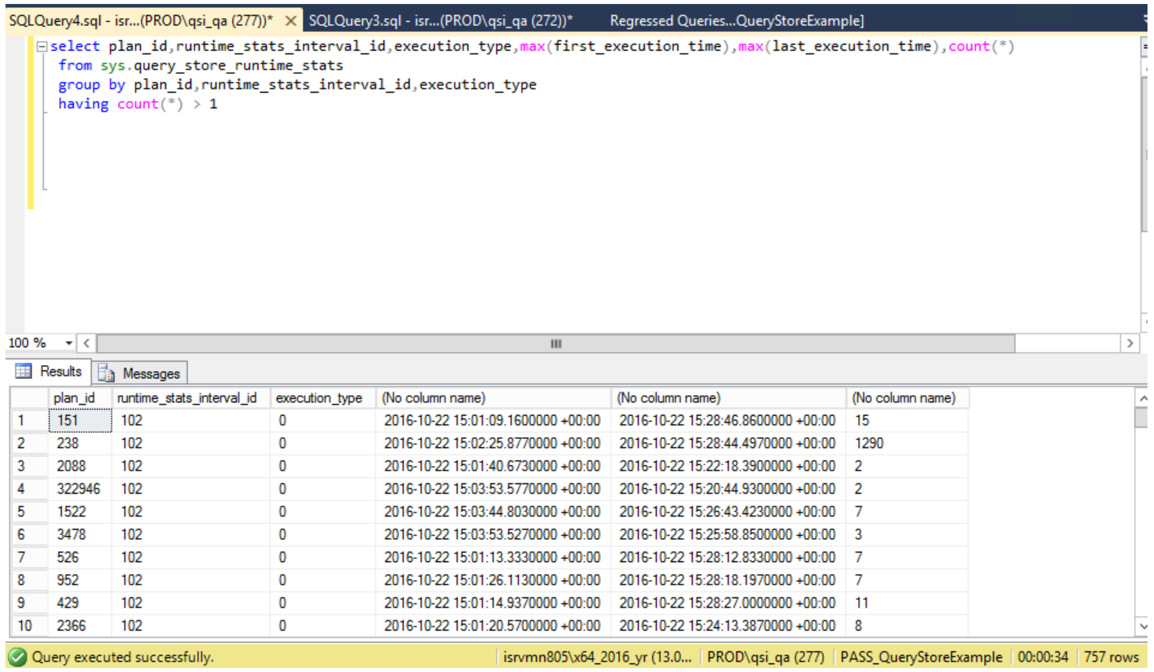
Runtime stats

- 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

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]

```sql
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

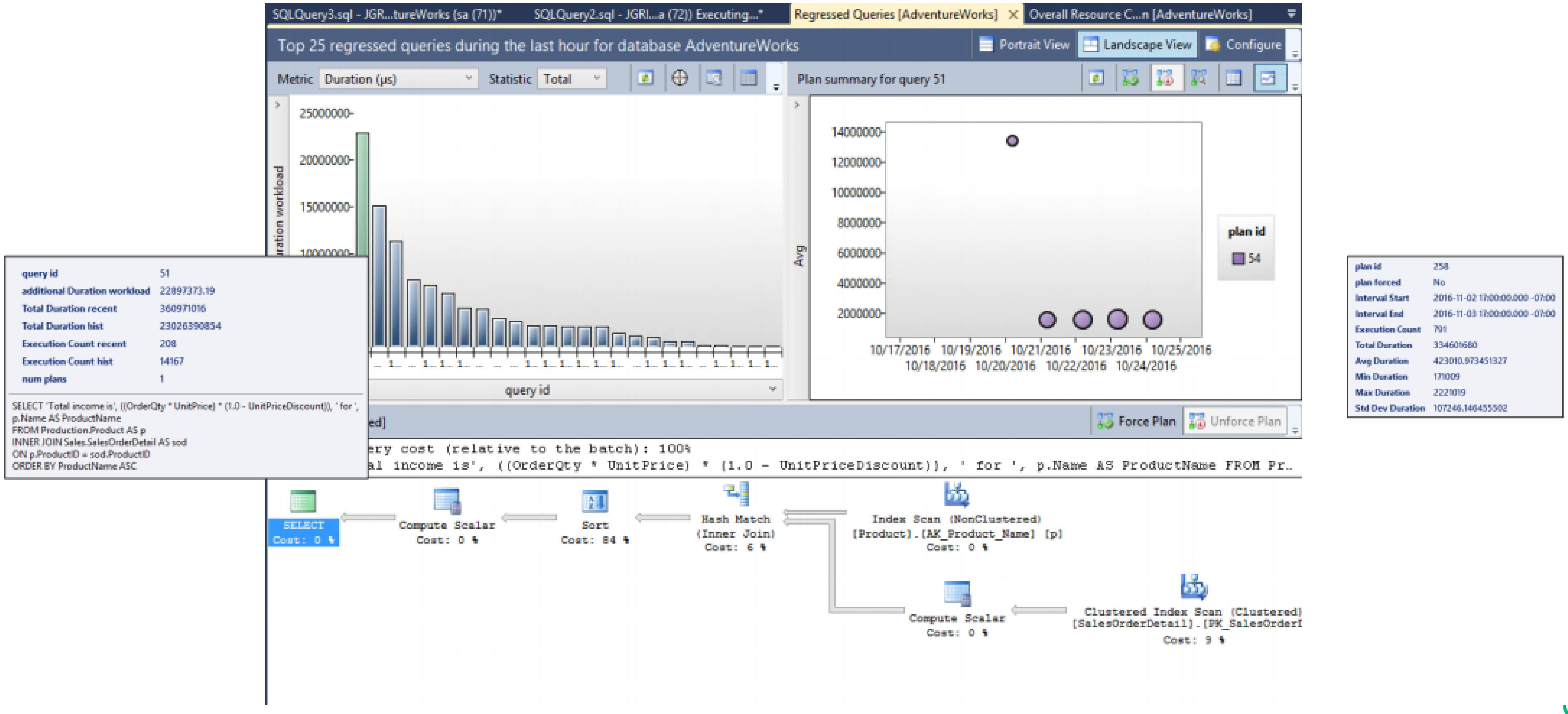
	plan_id	runtime_stats_interval_id	execution_type	(No column name)	(No column name)	(No column name)
1	151	102	0	2016-10-22 15:01:09.1600000 +00:00	2016-10-22 15:28:46.8600000 +00:00	15
2	238	102	0	2016-10-22 15:02:25.8770000 +00:00	2016-10-22 15:28:44.4970000 +00:00	1290
3	2088	102	0	2016-10-22 15:01:40.6730000 +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

Example Scenarios

Demo

Regressed Queries Report



Top 25 regressed queries during the last hour for database AdventureWorks

Metric: Duration (µs) Statistic: Total

query id	51
additional Duration workload	22897373.19
Total Duration recent	360971016
Total Duration hist	23026390854
Execution Count recent	208
Execution Count hist	14167
num plans	1

```
SELECT 'Total income is', ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName ASC
```

plan id	258
plan forced	No
Interval Start	2016-11-02 17:00:00.000 -07:00
Interval End	2016-11-03 17:00:00.000 -07:00
Execution Count	791
Total Duration	334601680
Avg Duration	423010.973451327
Min Duration	171009
Max Duration	2221019
Std Dev Duration	107246.146455502

```
ery cost (relative to the batch): 100%
al income is', ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)), ' for ', p.Name AS ProductName FROM Pr..
```

SELECT Cost: 0 %

Compute Scalar Cost: 0 %

Sort Cost: 84 %

Hash Match (Inner Join) Cost: 6 %

Index Scan (NonClustered) [Product].[AK_Product_Name] [p] Cost: 0 %

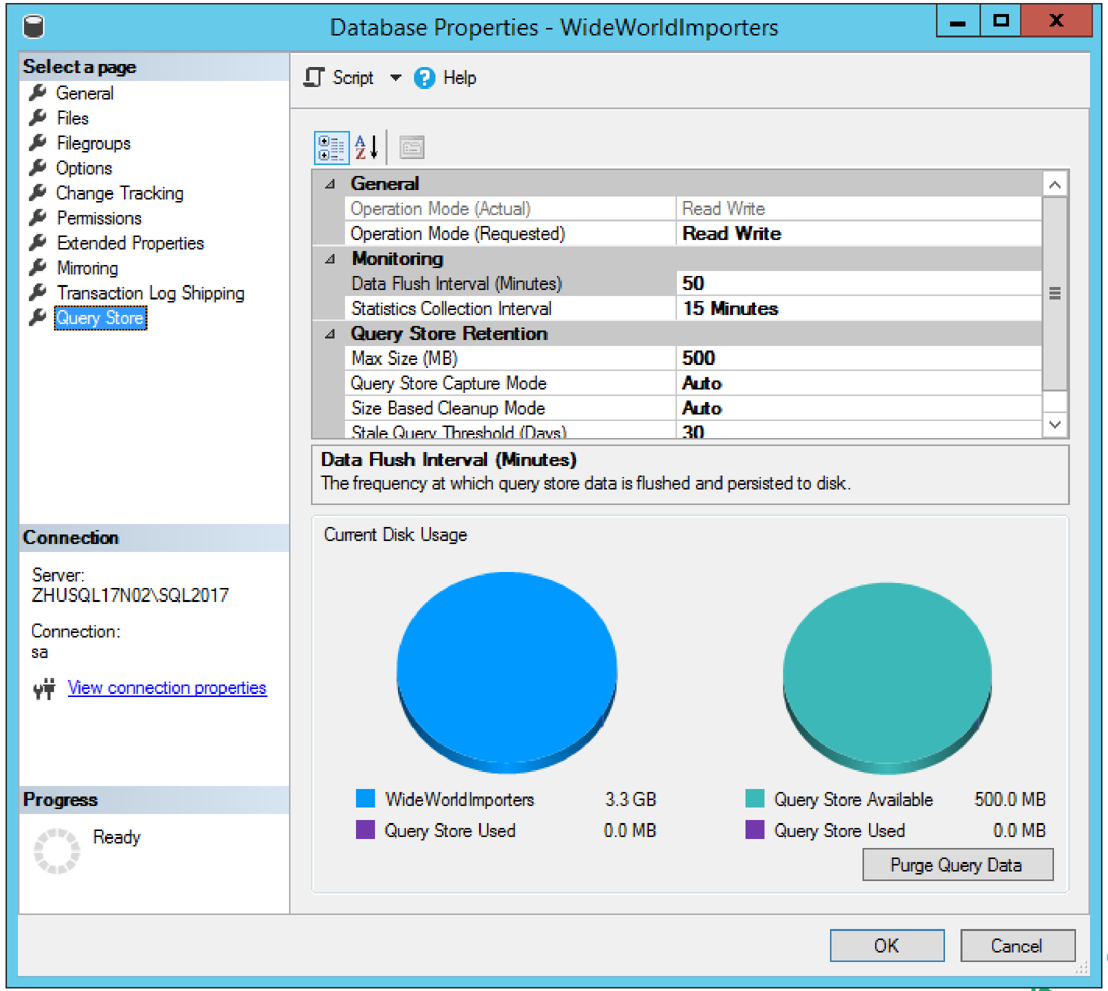
Compute Scalar Cost: 0 %

Clustered Index Scan (Clustered) [SalesOrderDetail].[PK_SalesOrderI Cost: 9 %

Configure Query Store

-- T-SQL

```
ALTER DATABASE PASS_QueryStoreExample SET
QUERY_STORE
(
OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30),
-- days to retain data in q-store
DATA_FLUSH_INTERVAL_SECONDS = 900,
-- frequency data is persisted to disk
INTERVAL_LENGTH_MINUTES = 60,                 -- interval to
aggregate runtime exec stats  MAX_STORAGE_SIZE_MB =
1024,                    -- maximum size of the q-store
QUERY_CAPTURE_MODE = ALL,                     -- type of
queries q-store captures (tracked/auto)
MAX_PLANS_PER_QUERY=5,                        -- maximum
plans maintained for each query.
SIZE_BASED_CLEANUP_MODE = AUTO                -- controls
the cleanup process
)
GO
```



Database Properties - WideWorldImporters

Select a page
- General
- Files
- Filegroups
- Options
- Change Tracking
- Permissions
- Extended Properties
- Mirroring
- Transaction Log Shipping
- Query Store

Script ▾ Help

General	
Operation Mode (Actual)	Read Write
Operation Mode (Requested)	**Read Write**
Monitoring	
Data Flush Interval (Minutes)	**50**
Statistics Collection Interval	**15 Minutes**
Query Store Retention	
Max Size (MB)	**500**
Query Store Capture Mode	**Auto**
Size Based Cleanup Mode	**Auto**
Stale Query Threshold (Days)	**30**

Data Flush Interval (Minutes)
The frequency at which query store data is flushed and persisted to disk.

Connection

Server:
ZHUSQL17N02\SQL2017

Connection:
sa

View connection properties

Progress

Ready

Current Disk Usage

🟦 WideWorldImporters	3.3 GB	🟩 Query Store Available	500.0 MB
🟪 Query Store Used	0.0 MB	🟪 Query Store Used	0.0 MB

Purge Query Data

OK Cancel

Default configuration

- **Operation Mode** : Off
- **Data Flush Interval (Minutes)** : 15 minutes
- **Statistics Collection Interval** : 1 hour
- **Maximum size allocated to the Query Store** : 100 MB
- **Query Store Capture Mode** : 'All'
- **Size Based Cleanup Mode size** : 2016 - 'Off', 2017 – 'Auto'
- **Stale Query Threshold (Days)** : 30 days
- **Plans Per Query** : 200 plans - Must change via T-SQL not in SSMS
- **Wait Stat Capture Mode**: 1

Management

- **sp_query_store_flush_db** - Flushes in-memory portion of data to disk
- **sp_query_store_reset_exec_stats plan_id** - Clears runtime stats for a specific query plan
- **sp_query_store_force_plan query_id, plan_id** - Enables forcing a particular plan
- **sp_query_store_unforce_plan query_id, plan_id** - Removes forcing a particular plan for a particular query
- **sp_query_store_remove_plan plan_id** - Removes a single plan from the Query Store
- **sp_query_store_remove_query query_id** - Removes a single query, all associated plans/ statistics from Query Store

Gotchas

- 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
- Query Store disabled by default
  - Consider enabling at model database to be available for each db
- No global configuration
  - Without manual scripting or setting the model DB
  - Multiple DBA's could change configuration - hard to track
  - Changes to QS are stored in the errorlog

Space Consumption Must be Managed

- Turns read-only when reach capacity
  - 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

How much disk space is required

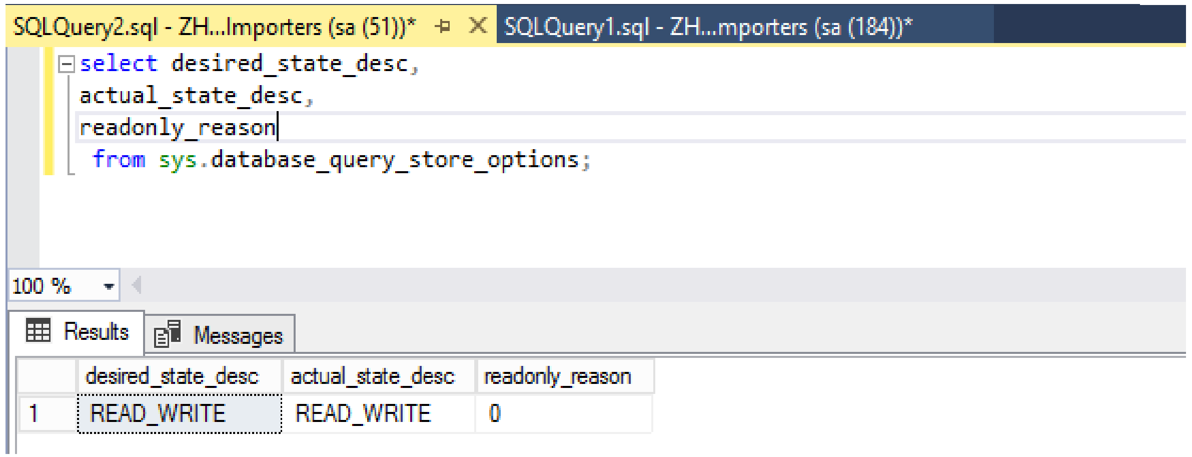
Configuration Parameters

- Retention period
- Type of queries and data to capture
- Plan per query

Database Workload

- Amount of different query texts
- Using non-parameterized queries

https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-database-query-store-options-transact-sql?view=sql-server-2017


SQLQuery2.sql - ZH...Importers (sa (51))*
SQLQuery1.sql - ZH...mporters (sa (184))*
100 %
Results
Messages

```
select desired_state_desc,
actual_state_desc,
readonly_reason
 from sys.database_query_store_options;
```

	desired_state_desc	actual_state_desc	readonly_reason
1	READ_WRITE	READ_WRITE	0

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.

More Gotchas

- Little context – who run a query? From what program? etc
- Allows forcing of a better plan for a query
  - Good as a quick **temporary** fix
  - Can be easily forgotten
  - Plan forcing can fail if outdated due to data and/or schema changes and more
- Plan and query texts contains literals and may contain PII data
- Can have an impact on performance

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
- Overall load

Query Store usage

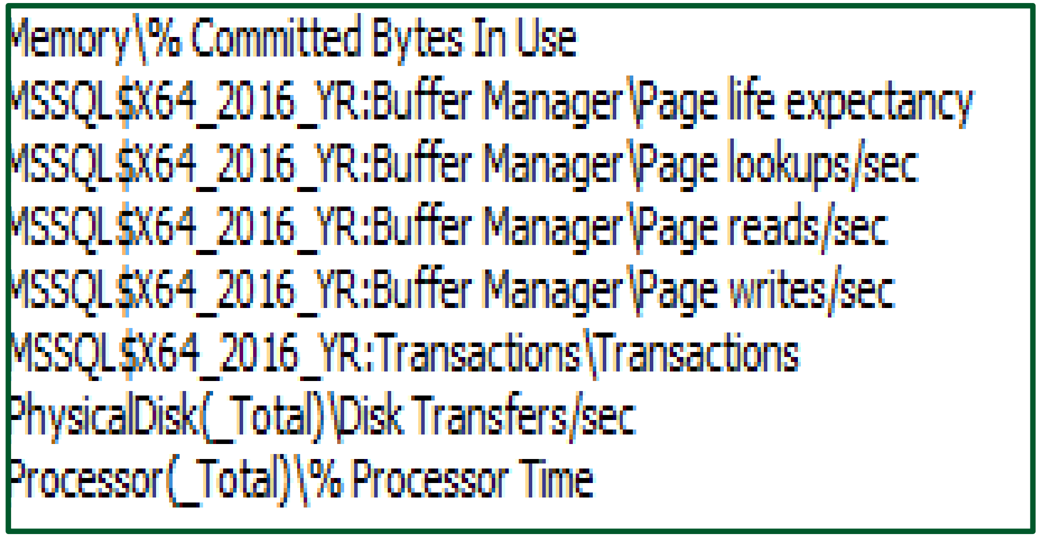
- Amount of queries run against the store and their type

Version

- Bug affecting performance - https://support.microsoft.com/en-us/help/4340759

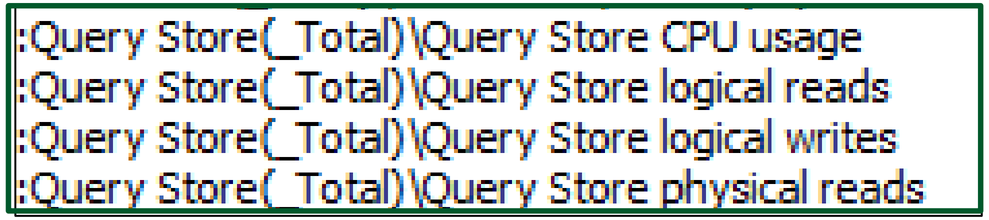
Monitor - Perfmon Counters

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
```

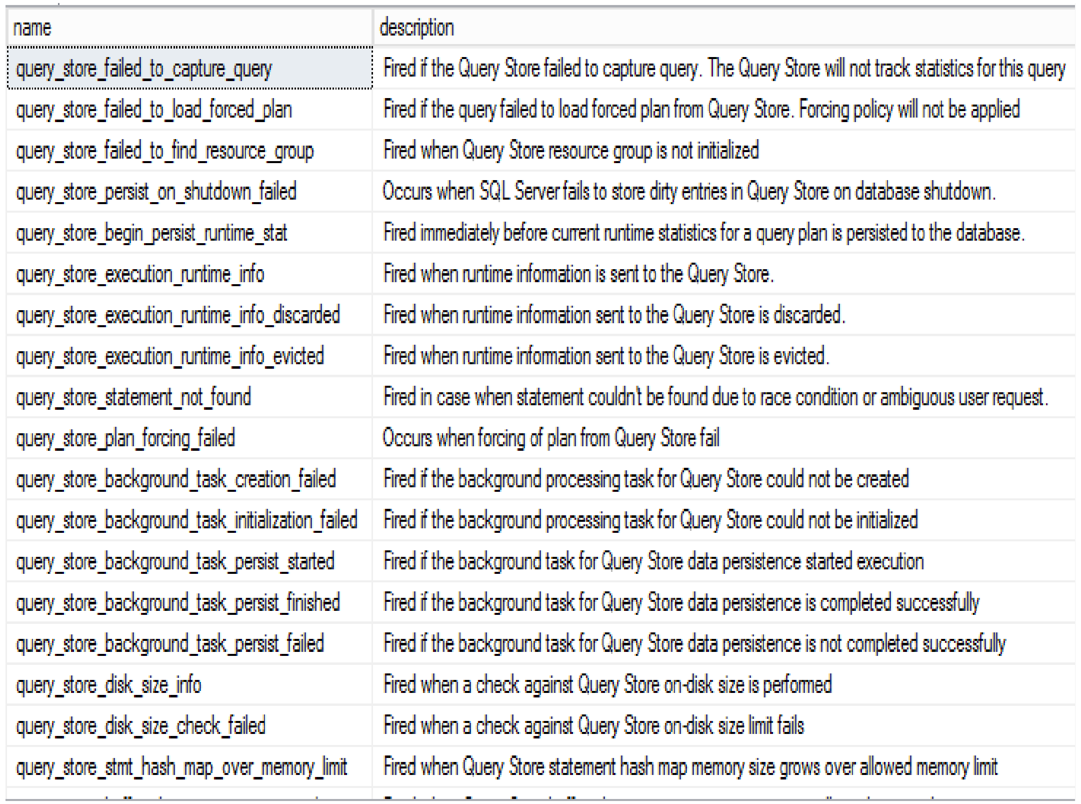
Query Store



```
: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
```

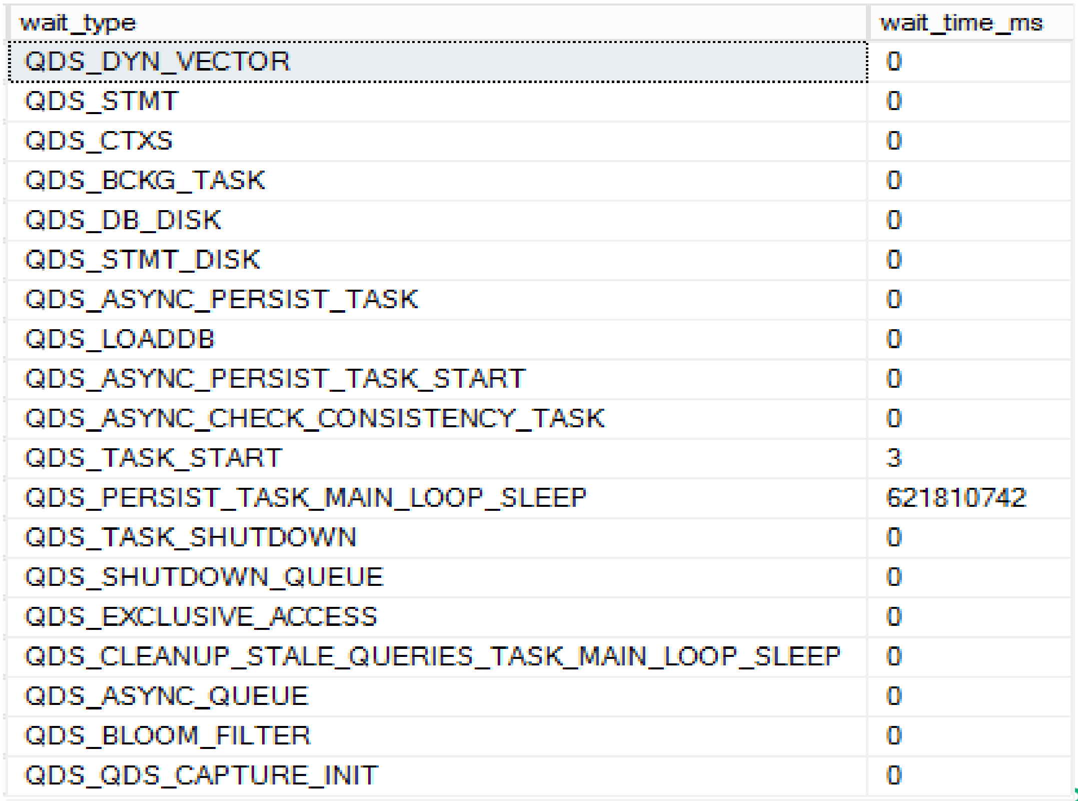
Monitor – Waits and extended events

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

What’s new in 2017

Wait Events

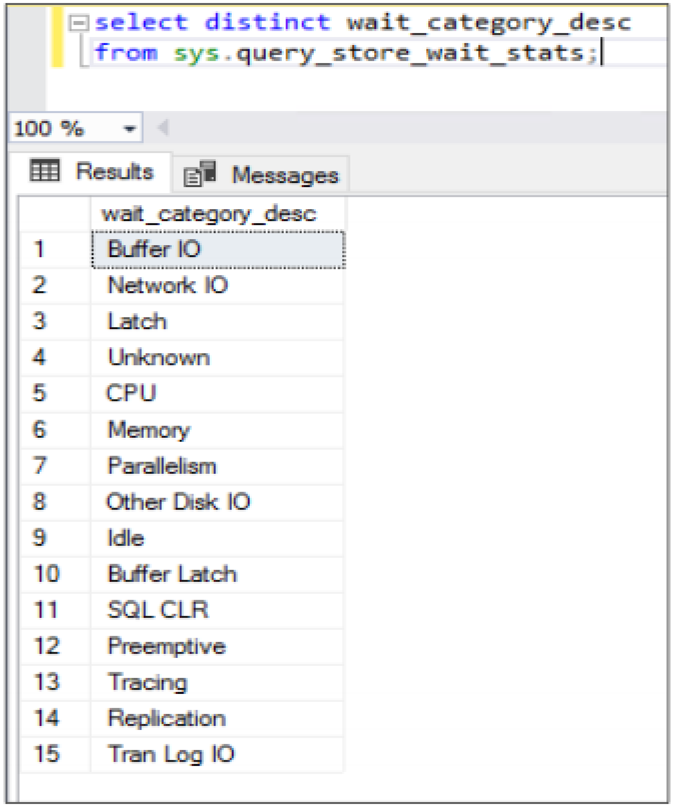
- Accumulated whenever query is not on the scheduler
- Important to understand what is going on:
  - SOS_Scheduler_Yield – wait for CPU
  - LCK* - Lock
  - CXPACKT – wait for parallel threads to finish
  - PAGEIOLATCH* – physical IO
  - PAGELATCH – logical io – wait for buffer page
  - ASYNC_NETWORK – send data to client over the network

- Pre 2017 – only available for running requests and on instance level (sys.dm_os_wait_stats)
- 2017 – also added more persistent information at session level (sys.dm_exec_session_wait_stats)

=> Not useful for historical analysis

Adding the Wait Stats store

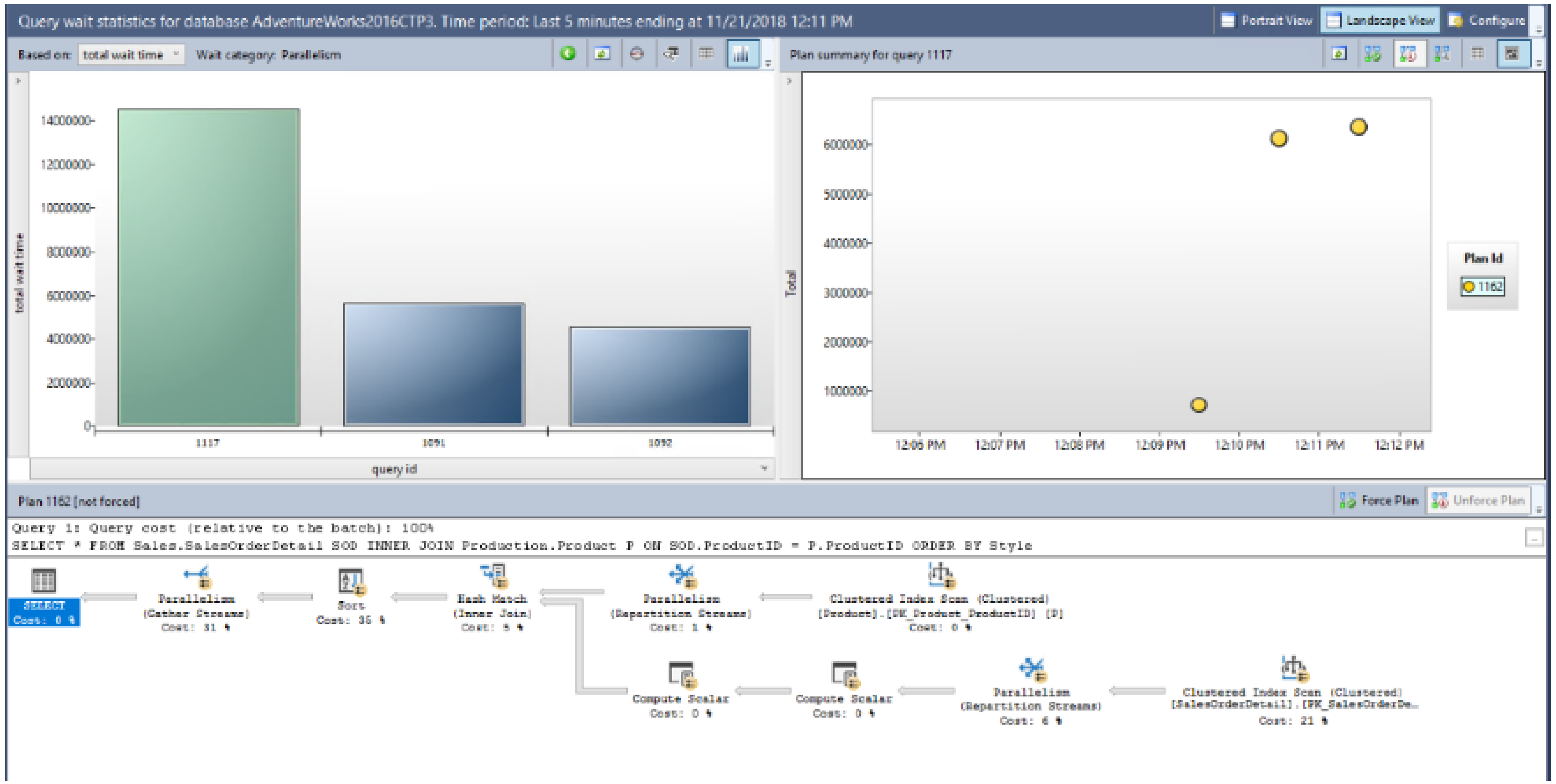
- New Wait Stats Store include with plan and runtime stats store at query level
- T-SQL: WAIT_STATS_CAPTURE_MODE
  - On = 1 (default)
  - OFF = 0
- Waits are mapped to 23 Wait Categories
  - Summarizes wait types
  - use based on similarity in response



```
select distinct wait_category_desc
from sys.query_store_wait_stats;
```

	wait_category_desc
1	Buffer IO
2	Network IO
3	Latch
4	Unknown
5	CPU
6	Memory
7	Parallelism
8	Other Disk IO
9	Idle
10	Buffer Latch
11	SQL CLR
12	Preemptive
13	Tracing
14	Replication
15	Tran Log IO

New SSMS view



Query wait statistics for database AdventureWorks2016CTP3. Time period: Last 5 minutes ending at 11/21/2018 12:11 PM

Based on: total wait time   Wait category: Parallelism

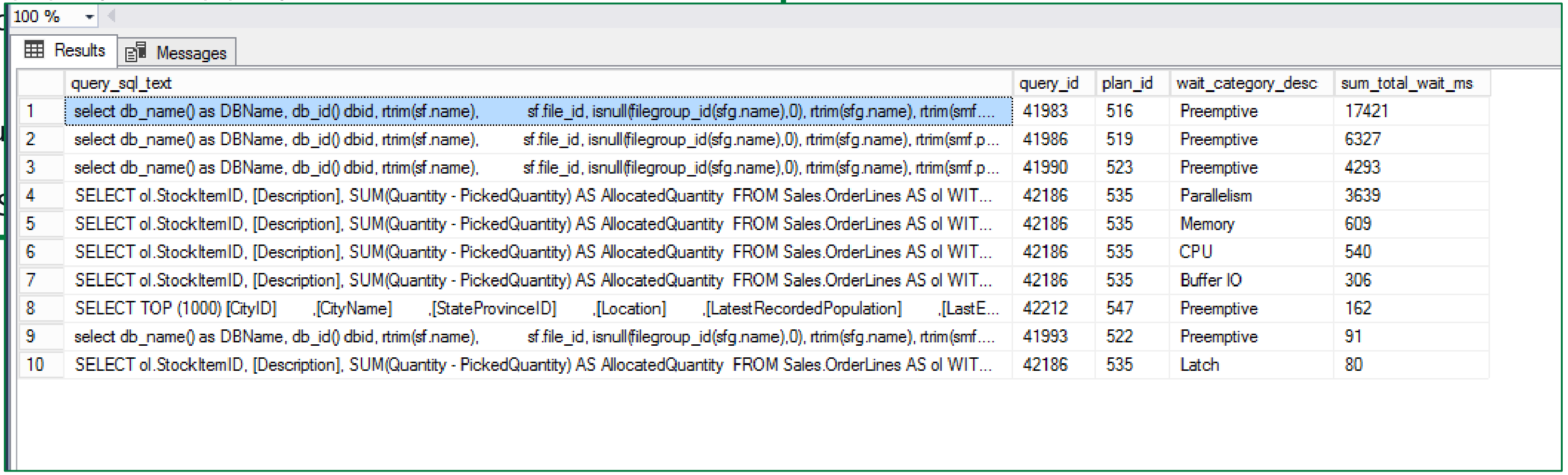
Plan summary for query 1117

Plan 1162 [not forced]

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Sales.SalesOrderDetail SOD INNER JOIN Production.Product P ON SOD.ProductID = P.ProductID ORDER BY Style
```

Show top 10 queries

```
SELECT TOP 10
qt.query_sql_text,
q.query_id,
p.plan_id,
ws.wait_category_desc,
sum(total_query_wait_time_ms) AS sum_total_wait_ms
FROM sys.query_store_wait_stats ws
JOIN sys.query_store_plan p ON ws.plan_id = p.plan_id
JOIN sys.query_store_query q ON p.query_id = q.query_id
JOIN sys.query_store_query_text q
qt.query_text_id
WHERE ws.wait_category_desc !=
GROUP BY qt.query_sql_text, q.qu
ws.wait_category_desc
ORDER BY sum_total_wait_ms DES
```



	query_sql_text	query_id	plan_id	wait_category_desc	sum_total_wait_ms
1	select db_name() as DBName, db_id() dbid, rtrim(sf.name), sf.file_id, isnull(filegroup_id(sfg.name),0), rtrim(sfg.name), rtrim(smf....	41983	516	Preemptive	17421
2	select db_name() as DBName, db_id() dbid, rtrim(sf.name), sf.file_id, isnull(filegroup_id(sfg.name),0), rtrim(sfg.name), rtrim(smf.p...	41986	519	Preemptive	6327
3	select db_name() as DBName, db_id() dbid, rtrim(sf.name), sf.file_id, isnull(filegroup_id(sfg.name),0), rtrim(sfg.name), rtrim(smf.p...	41990	523	Preemptive	4293
4	SELECT ol.StockItemID, [Description], SUM(Quantity - PickedQuantity) AS AllocatedQuantity FROM Sales.OrderLines AS ol WIT...	42186	535	Parallelism	3639
5	SELECT ol.StockItemID, [Description], SUM(Quantity - PickedQuantity) AS AllocatedQuantity FROM Sales.OrderLines AS ol WIT...	42186	535	Memory	609
6	SELECT ol.StockItemID, [Description], SUM(Quantity - PickedQuantity) AS AllocatedQuantity FROM Sales.OrderLines AS ol WIT...	42186	535	CPU	540
7	SELECT ol.StockItemID, [Description], SUM(Quantity - PickedQuantity) AS AllocatedQuantity FROM Sales.OrderLines AS ol WIT...	42186	535	Buffer IO	306
8	SELECT TOP (1000) [CityID] ,[CityName] ,[StateProvinceID] ,[Location] ,[LatestRecordedPopulation] ,[LastE...	42212	547	Preemptive	162
9	select db_name() as DBName, db_id() dbid, rtrim(sf.name), sf.file_id, isnull(filegroup_id(sfg.name),0), rtrim(sfg.name), rtrim(smf....	41993	522	Preemptive	91
10	SELECT ol.StockItemID, [Description], SUM(Quantity - PickedQuantity) AS AllocatedQuantity FROM Sales.OrderLines AS ol WIT...	42186	535	Latch	80

Azure Data Studio (operations studio)

- Cross-platform database tool for data professionals
- Supports on-premises and cloud data platforms on Windows, MacOS, and Linux.
- Runs on Windows, macOS, and Linux.
- Download from https://docs.microsoft.com/en-us/sql/azure-data-studio/download?view=sql-server-2017
- Demo

Best Practice

Best Practice

- Plan Ahead
  - Consider for which use case
  - Consider for which databases
  - Turn on when needed at the level needed
- Maintain
  - Size according to workload
  - Consider how long to keep it
  - Backups & Restore
  - Create standard setup & configuration scripts
  - Install latest CU - https://www.sqlskills.com/blogs/erin/important-query-store-fixes-january-2019/
- Monitor
  - Size and status
  - Performance
  - Failed forced plans

Summary

- Excellent to find out :
  - what happened (flight recorder)
  - what changed
  - what is going on – top consumer, waits
- Allow performance monitoring for all
- Plan to use it with proper impact monitoring

Reference

- 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/
- https://docs.microsoft.com/en-us/sql/relational-databases/performance/monitoring-performance-by-using-the-query-store?view=sql-server-2017
- file:///C:/Users/adror/Downloads/PASS_Whats_new_in_The_2017_Query_Store%20(1).pdf
- https://www.sqlskills.com/blogs/erin/query-store-performance-overhead/
- https://docs.microsoft.com/en-us/sql/azure-data-studio/tutorial-build-custom-insight-sql-server?view=sql-server-2017

Our generous sponsors

Global SQLSaturday Partner

Microsoft Azure

PASS

Gold Sponsor

EXPERDA

Quest

elastic

BARRY KATZ
Communicating Success

Couchbase

Silver Sponsor

DATASITE

VALINOR
YOUR DATA. OUR EXPERTS.

actifio

Bronze Sponsor

inspire
TECHNOLOGIES

Session and Event Evaluations

Your feedback is important and valuable.

2

Links to follow

Session evaluation: https://www.sqlsaturday.com/823/Sessions/SessionEvaluation.aspx

Event evaluation: https://www.sqlsaturday.com/823/EventEval.aspx

Appendix

examples

Find Queries with multiple plans

```
;WITH Query_MultPlans
AS
(
SELECT COUNT(*) AS cnt, q.query_id
FROM sys.query_store_query_text AS qt
JOIN sys.query_store_query AS q
ON qt.query_text_id = q.query_text_id
JOIN sys.query_store_plan AS p
ON p.query_id = q.query_id
GROUP BY q.query_id
HAVING COUNT(distinct plan_id) > 1
)
SELECT q.query_id, object_name(object_id) AS ContainingObject,
query_sql_text, plan_id,-- convert(xml,p.query_plan) AS plan_xml,
p.last_compile_start_time, p.last_execution_time
FROM Query_MultPlans AS qm
JOIN sys.query_store_query AS q
ON qm.query_id = q.query_id
JOIN sys.query_store_plan AS p
ON q.query_id = p.query_id
JOIN sys.query_store_query_text qt
ON qt.query_text_id = q.query_text_id
ORDER BY query_id, plan_id;
```

Find Queries that aborted during last week

```
select  qt.query_sql_text,  q.query_hash,
q.query_id, qt.query_text_id, p.plan_id,sum(count_executions) as executions#
from
sys.query_store_query_text AS qt
JOIN sys.query_store_query AS q
    ON qt.query_text_id = q.query_text_id
JOIN sys.query_store_plan AS p
    ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats AS rs
    ON p.plan_id = rs.plan_id
JOIN sys.query_store_runtime_stats_interval si
    ON si.runtime_stats_interval_id=rs.runtime_stats_interval_id
where execution_type <> 0                              --only aborted executions
and rs.last_execution_time >= DATEADD(DAY, -7, GETUTCDATE())
group by qt.query_sql_text,   q.query_hash,
    q.query_id, qt.query_text_id, p.plan_id
```

Find Queries lack Parameterization

same query hash for multiple queries with the same plan

```
select qs.query_hash,count_queries,count_plans
from
(select query_hash,count(q.query_id) as count_queries
from sys.query_store_query q join
 sys.query_store_plan p
on q.query_id = p.query_id
group by query_hash) qs
join
(select query_hash,count(query_plan_hash) as count_plans
 from
( select distinct query_hash,query_plan_hash
  from sys.query_store_query q
  join sys.query_store_plan p
  on q.query_id = p.query_id
) as qp1
group by query_hash
) qp
on qs.query_hash=qp.query_hash
where count_queries > 10 and count_plans < 10
```

```
-- find text of a query based on its hash
select query_sql_text,query_plan_hash
from
sys.query_store_query q join
 sys.query_store_plan p
on q.query_id = p.query_id
join
sys.query_store_query_text t on
q.query_text_id=t.query_text_id
where query_hash=<query hash>
```

Find Long Runing Queries in last Hour

```
;WITH AggregatedDurationLastHour
AS
(
SELECT q.query_id, SUM(count_executions * avg_duration) AS total_duration,
COUNT (distinct p.plan_id) AS number_of_plans
FROM sys.query_store_query_text AS qt JOIN sys.query_store_query AS q
ON qt.query_text_id = q.query_text_id
JOIN sys.query_store_plan AS p ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats AS rs ON rs.plan_id = p.plan_id
JOIN sys.query_store_runtime_stats_interval AS rsi
ON rsi.runtime_stats_interval_id = rs.runtime_stats_interval_id
WHERE rsi.start_time >= DATEADD(hour, -1, GETUTCDATE())
AND rs.execution_type_desc = 'Regular'
GROUP BY q.query_id
) ,OrderedDuration
AS
(
SELECT query_id, total_duration, number_of_plans,
ROW_NUMBER () OVER (ORDER BY total_duration DESC, query_id) AS RN
FROM AggregatedDurationLastHour
)
SELECT qt.query_sql_text, object_name(q.object_id) AS
containing_object, q.query_id,
p.plan_id,rsi.start_time as interval_start, rs.avg_duration,rs.count_executions,
CONVERT(xml, p.query_plan) AS query_plan_xml
FROM OrderedDuration od JOIN sys.query_store_query
AS q ON q.query_id = od.query_id
JOIN sys.query_store_query_text AS qt ON
q.query_text_id = qt.query_text_id
JOIN sys.query_store_plan AS p ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats AS rs ON rs.plan_id = p.plan_id
JOIN sys.query_store_runtime_stats_interval AS rsi ON rsi.runtime_stats_interval_id = rs.runtime_stats_interval_id
WHERE rsi.start_time >= DATEADD(hour, -1,GETUTCDATE())
AND number_of_plans > 1
ORDER BY total_duration DESC, query_id,
rsi.runtime_stats_interval_id, p.plan_id
```