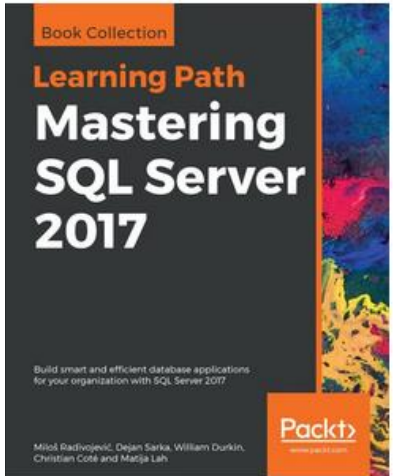
INTRODUCTION TO QUERY STORE

SQLPORT 116, ONLINE MEETING

MILOŠ RADIVOJEVIĆ, PRINCIPAL DATABASE CONSULTANT, BWIN GVC, AUSTRIA

ABOUT ME

- MILOŠ RADIVOJEVIĆ
- Data Platform MVP
- Principal Database Consultant at bwin, Vienna, Austria
- Co-Founder: SQL Pass Austria
- Conference Speaker, Book Author



Book Collection

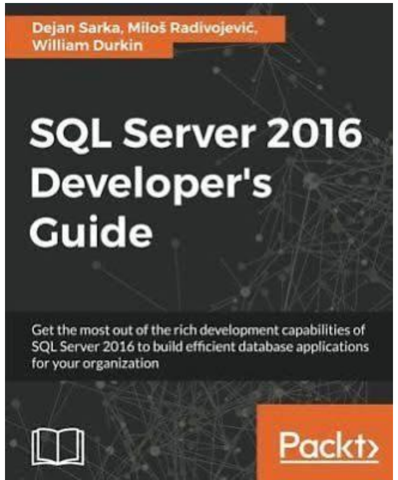
Learning Path

Mastering SQL Server 2017

Build smart and efficient database applications for your organization with SQL Server 2017

Miloš Radivojević, Dejan Sarka, William Durkin, Christian Coté and Matija Lah

Packt>

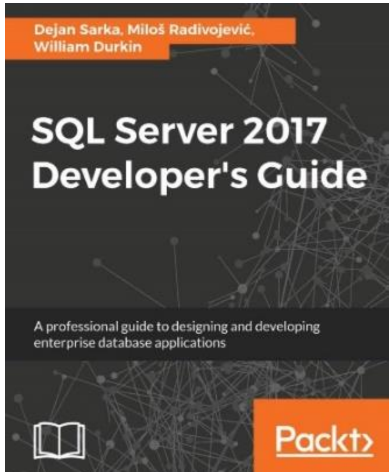


Dejan Sarka, Miloš Radivojević, William Durkin

SQL Server 2016 Developer's Guide

Get the most out of the rich development capabilities of SQL Server 2016 to build efficient database applications for your organization

Packt>



Dejan Sarka, Miloš Radivojević, William Durkin

SQL Server 2017 Developer's Guide

A professional guide to designing and developing enterprise database applications

Packt>

AGENDA

- Life before Query Store
- Why Query Store?
- Query Store Architecture and Configuration
- Query Store in Action

PROBLEMS WITH DMV TROUBLESHOOTING

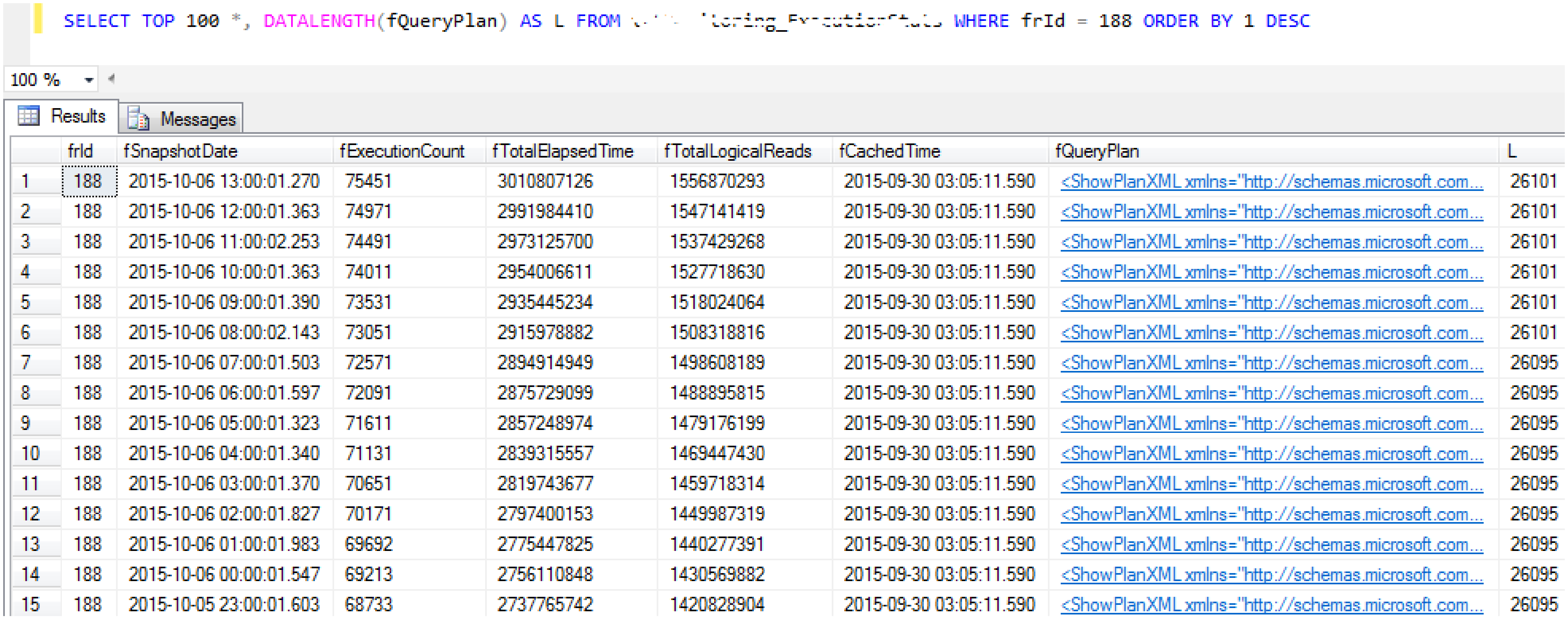
- reflects particular or aggregated information from the last server restart only
- when server goes down, cache disappears
- only the latest execution plan for a query is available
- you don’t know if and how execution plan changed over time

MY QUERY STORY

MY QUERY STORY

- Special stored procedure
  - Collect exec. parameters for each stored procedure (total elapsed time, total logical reads) within a database
  - Collect all different execution plans (XML) for the most important stored procedures
  - Store collected results in tables within a database
- SQL Job
- Call the special SP every 5 minutes

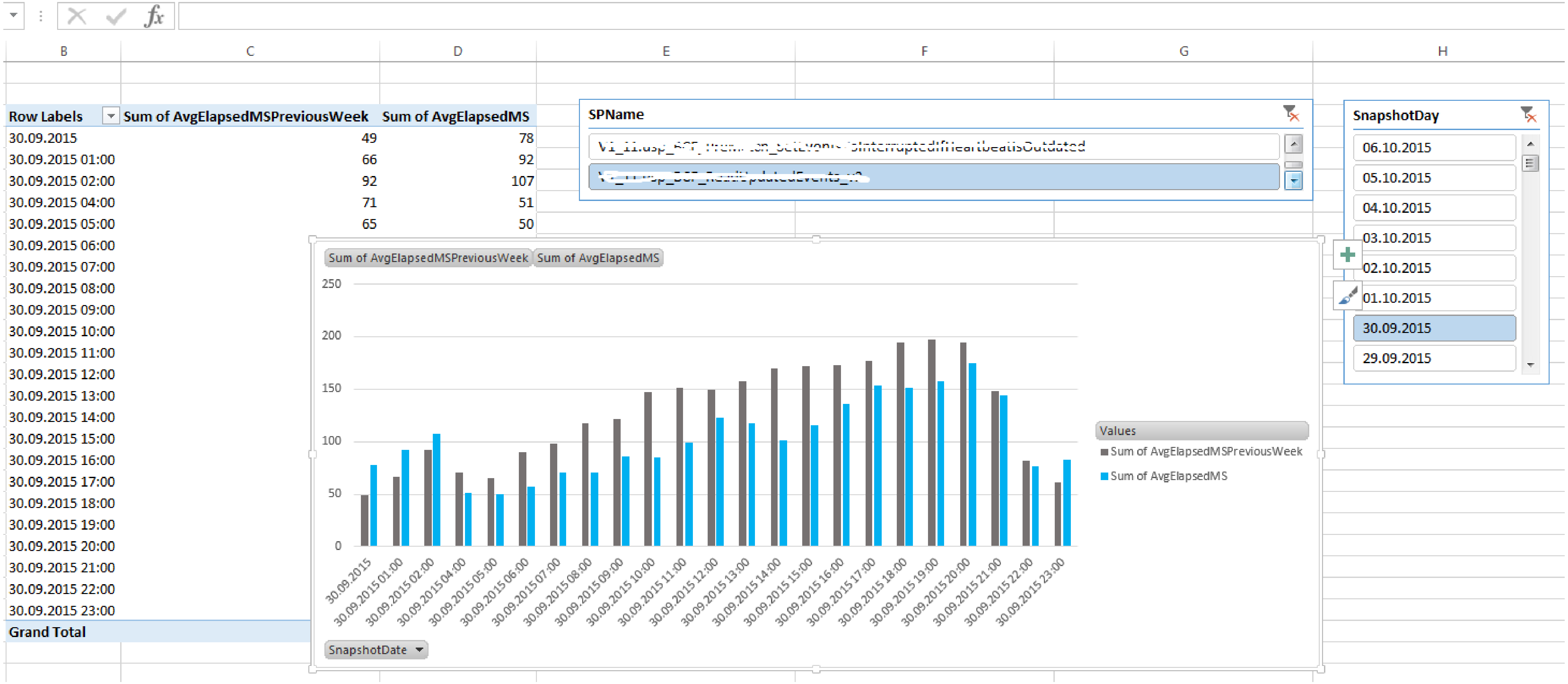
MY QUERY STORY



```
SELECT TOP 100 *, DATALENGTH(fQueryPlan) AS L FROM [redacted] WHERE frId = 188 ORDER BY 1 DESC
```

	frId	fSnapshotDate	fExecutionCount	fTotalElapsedTime	fTotalLogicalReads	fCachedTime	fQueryPlan	L
1	188	2015-10-06 13:00:01.270	75451	3010807126	1556870293	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26101
2	188	2015-10-06 12:00:01.363	74971	2991984410	1547141419	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26101
3	188	2015-10-06 11:00:02.253	74491	2973125700	1537429268	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26101
4	188	2015-10-06 10:00:01.363	74011	2954006611	1527718630	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26101
5	188	2015-10-06 09:00:01.390	73531	2935445234	1518024064	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26101
6	188	2015-10-06 08:00:02.143	73051	2915978882	1508318816	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26101
7	188	2015-10-06 07:00:01.503	72571	2894914949	1498608189	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
8	188	2015-10-06 06:00:01.597	72091	2875729099	1488895815	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
9	188	2015-10-06 05:00:01.323	71611	2857248974	1479176199	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
10	188	2015-10-06 04:00:01.340	71131	2839315557	1469447430	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
11	188	2015-10-06 03:00:01.370	70651	2819743677	1459718314	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
12	188	2015-10-06 02:00:01.827	70171	2797400153	1449987319	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
13	188	2015-10-06 01:00:01.983	69692	2775447825	1440277391	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
14	188	2015-10-06 00:00:01.547	69213	2756110848	1430569882	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095
15	188	2015-10-05 23:00:01.603	68733	2737765742	1420828904	2015-09-30 03:05:11.590	<ShowPlanXML xmlns="http://schemas.microsoft.com...	26095

MY QUERY STORY



Row Labels	Sum of AvgElapsedMSPreviousWeek	Sum of AvgElapsedMS
30.09.2015	49	78
30.09.2015 01:00	66	92
30.09.2015 02:00	92	107
30.09.2015 04:00	71	51
30.09.2015 05:00	65	50
30.09.2015 06:00		
30.09.2015 07:00		
30.09.2015 08:00		
30.09.2015 09:00		
30.09.2015 10:00		
30.09.2015 11:00		
30.09.2015 12:00		
30.09.2015 13:00		
30.09.2015 14:00		
30.09.2015 15:00		
30.09.2015 16:00		
30.09.2015 17:00		
30.09.2015 18:00		
30.09.2015 19:00		
30.09.2015 20:00		
30.09.2015 21:00		
30.09.2015 22:00		
30.09.2015 23:00		
Grand Total		

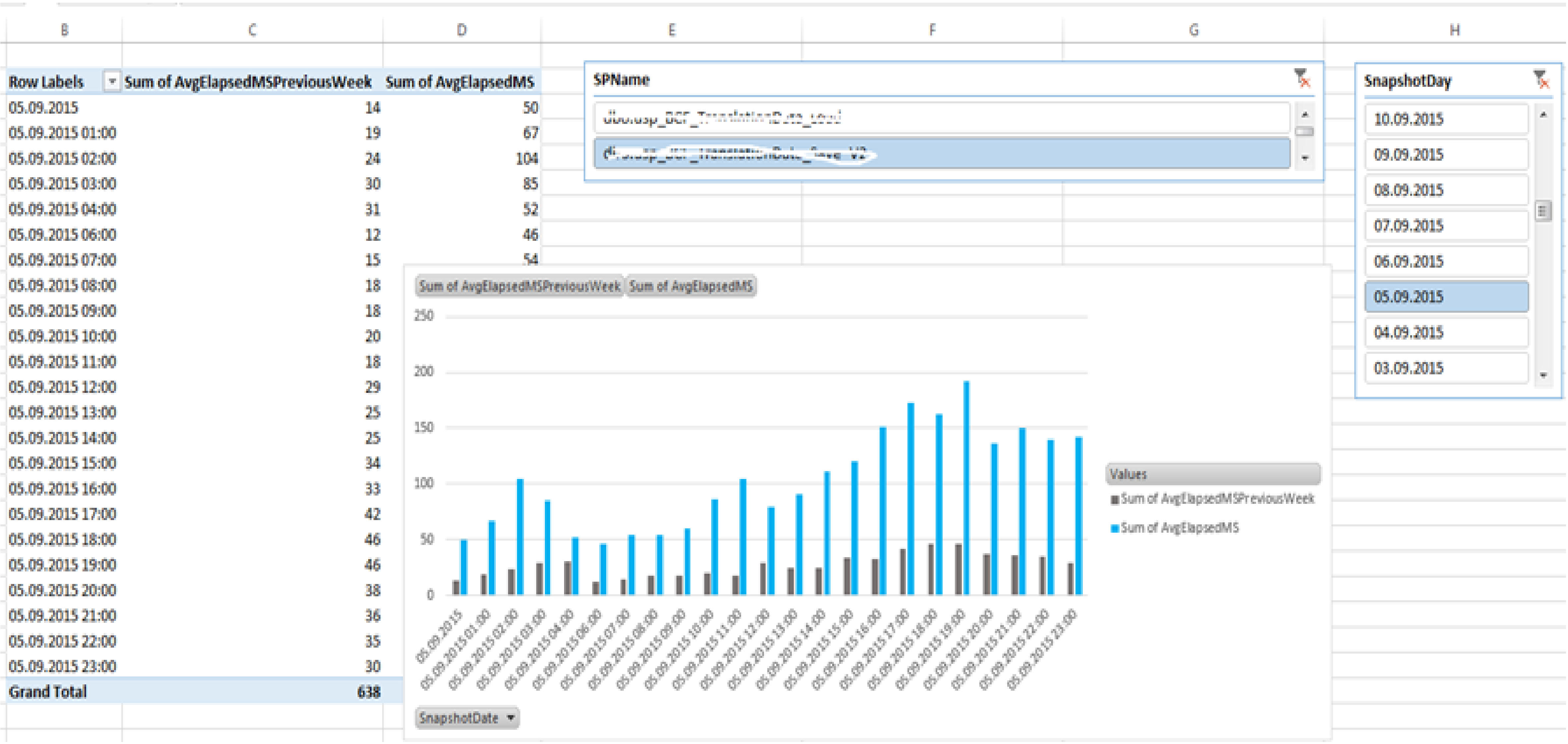
SPName

- [redacted]
- [redacted]

SnapshotDay

- 06.10.2015
- 05.10.2015
- 04.10.2015
- 03.10.2015
- 02.10.2015
- 01.10.2015
- 30.09.2015
- 29.09.2015

MY QUERY STORY



Row Labels	Sum of AvgElapsedMSPreviousWeek	Sum of AvgElapsedMS
05.09.2015	14	50
05.09.2015 01:00	19	67
05.09.2015 02:00	24	104
05.09.2015 03:00	30	85
05.09.2015 04:00	31	52
05.09.2015 06:00	12	46
05.09.2015 07:00	15	54
05.09.2015 08:00	18	
05.09.2015 09:00	18	
05.09.2015 10:00	20	
05.09.2015 11:00	18	
05.09.2015 12:00	29	
05.09.2015 13:00	25	
05.09.2015 14:00	25	
05.09.2015 15:00	34	
05.09.2015 16:00	33	
05.09.2015 17:00	42	
05.09.2015 18:00	46	
05.09.2015 19:00	46	
05.09.2015 20:00	38	
05.09.2015 21:00	36	
05.09.2015 22:00	35	
05.09.2015 23:00	30	
Grand Total	638	

SPName

[redacted]

[redacted]

SnapshotDay

10.09.2015

09.09.2015

08.09.2015

07.09.2015

06.09.2015

05.09.2015

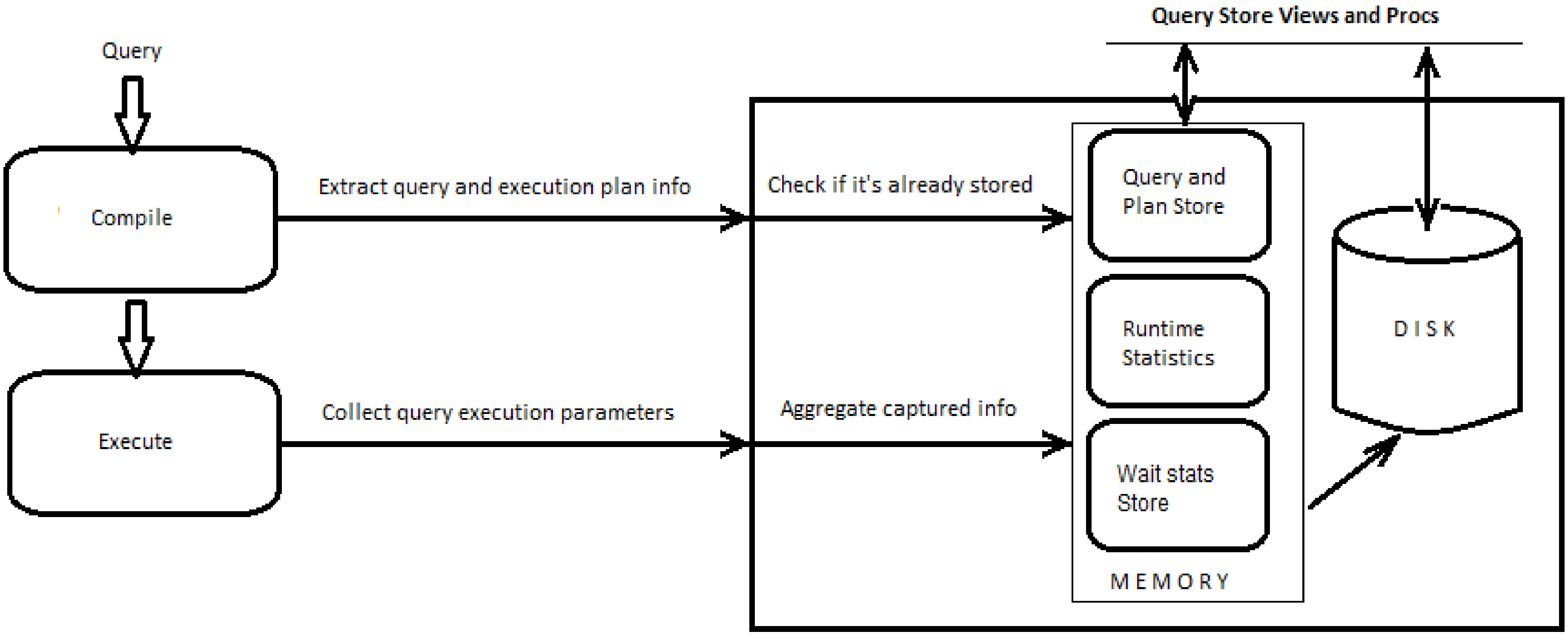
04.09.2015

03.09.2015

WHAT DOES THE QUERY STORE?

- Stores the history of queries, plans, execution details and waits statistics
  - Stores also plans with OPTION (RECOMPILE)
- Belongs to database
  - it is persistent – survives after restart, failover etc.
- Introduced with SQL Server 2016
- Disabled by default in on-prem databases, in SQL Azure is enabled
- Available in all editions (a few Enterprise features)
- Requires VIEW_DATABASE_STATE permission
- Cannot be enabled for system databases

QUERY STORE ARCHITECTURE


Query Store Views and Procs
Query
Compile
Extract query and execution plan info
Check if it's already stored
Query and Plan Store
Runtime Statistics
DISK
Execute
Collect query execution parameters
Aggregate captured info
Wait stats Store
MEMORY

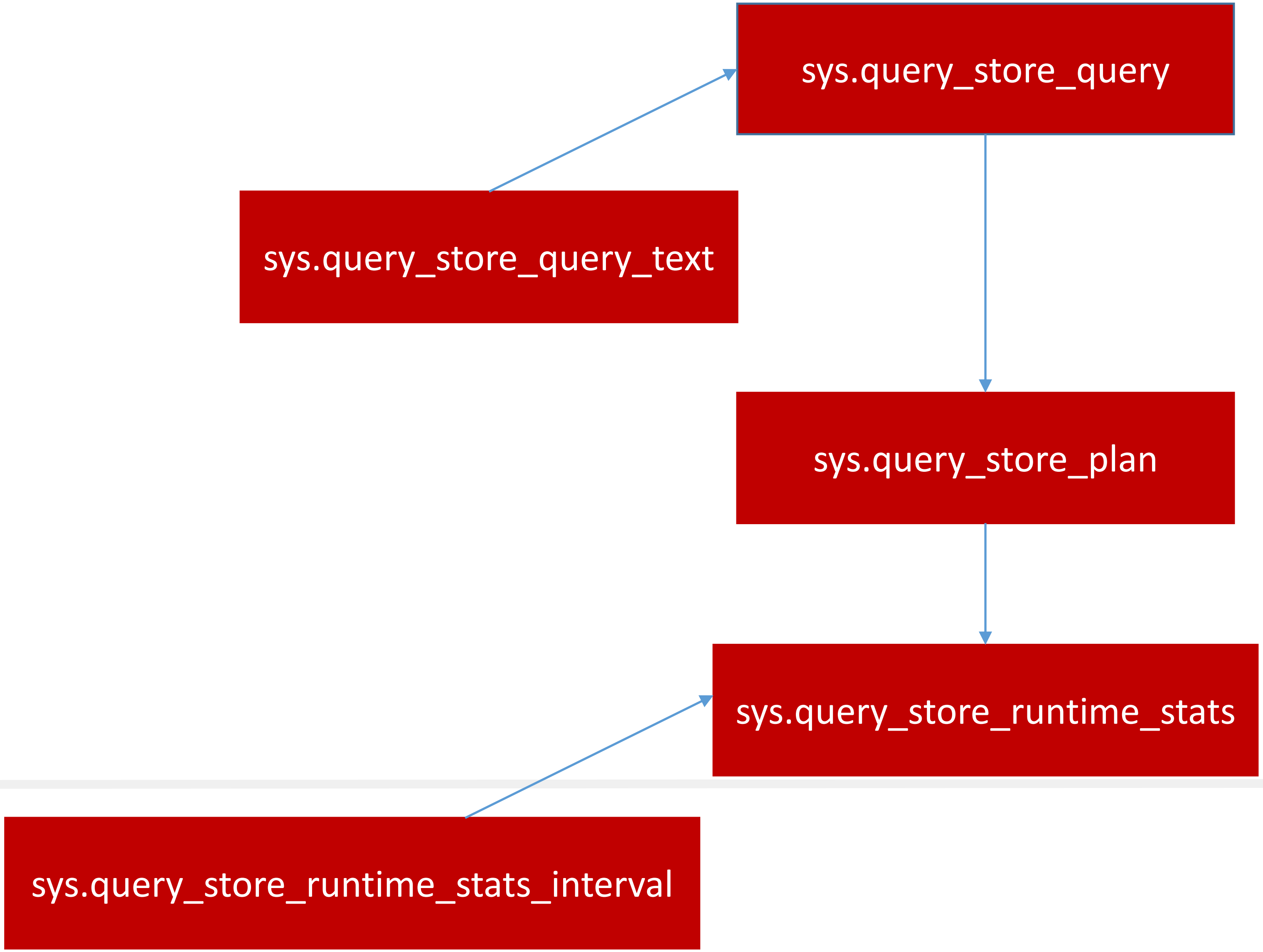
QUERY STORE

Query Text

Query
(id, object_id, compile details)

Execution Plans

Execution Parameters
(duration, CPU time, logical reads...)



sys.query_store_query

sys.query_store_query_text

sys.query_store_plan

sys.query_store_runtime_stats

sys.query_store_runtime_stats_interval

WHAT CAN WE DO WITH QUERY STORE?

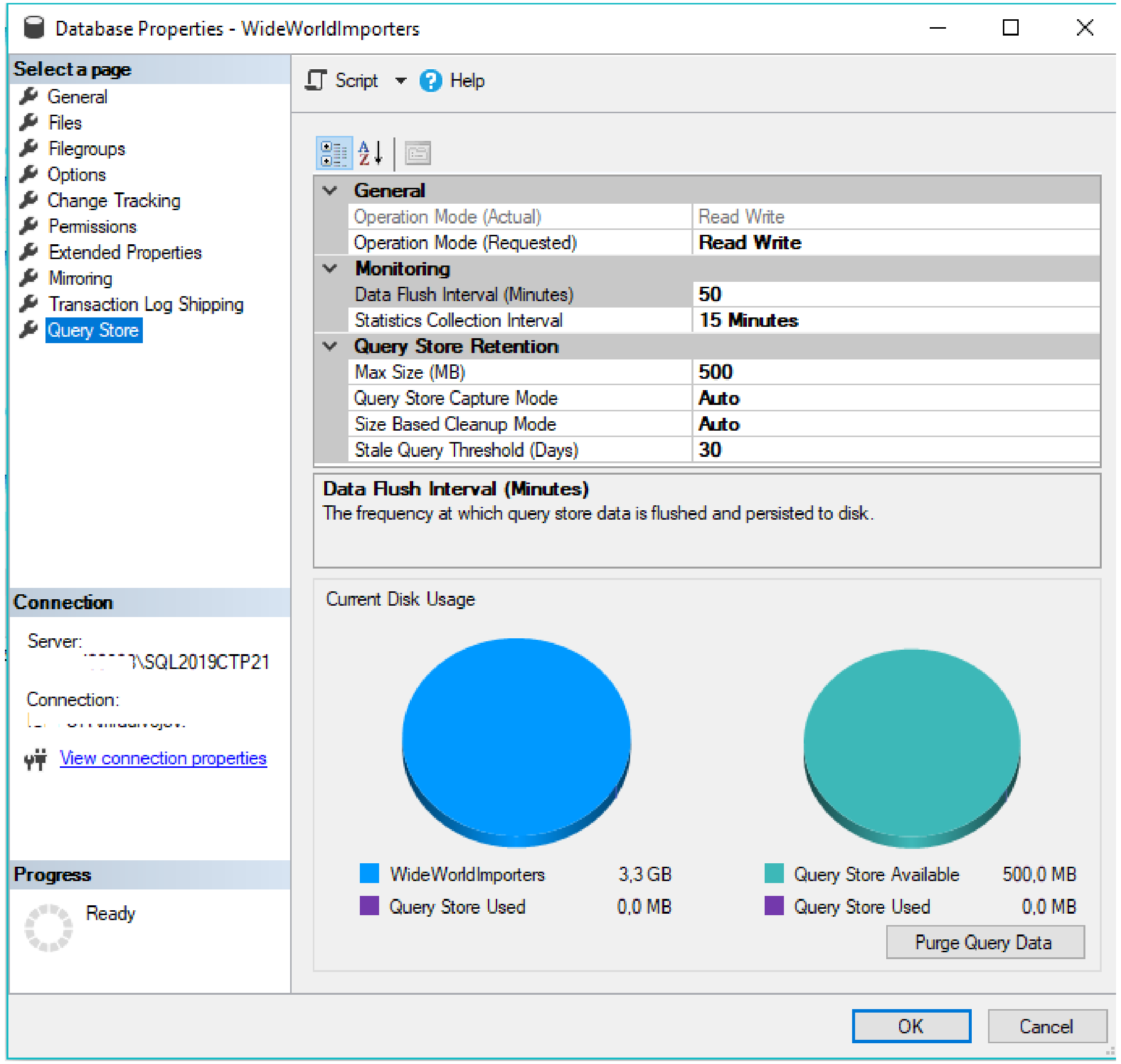
- Identifying query plan regressions
- Fixing plan regression (by forcing plans)
- Reducing the risk of upgrading, patching and reconfiguring
- Analyzing workload patterns
- Understanding waits at the plan level
- Identifying the most expensive and queries degraded over time

WHAT CAN WE DO WITH QUERY STORE?

- We can have answers to the following questions:
  - Was this query slow last weekend?
  - Why my query was slow last Saturday?
  - What are unstable queries (with multiple plans)?
  - We have some timeouts in the application. Are they from database?
  - Find out unfinished queries or queries that ended with an exception

CONFIGURING QUERY STORE

```
ALTER DATABASE WideWorldImporters SET QUERY_STORE = ON;
```



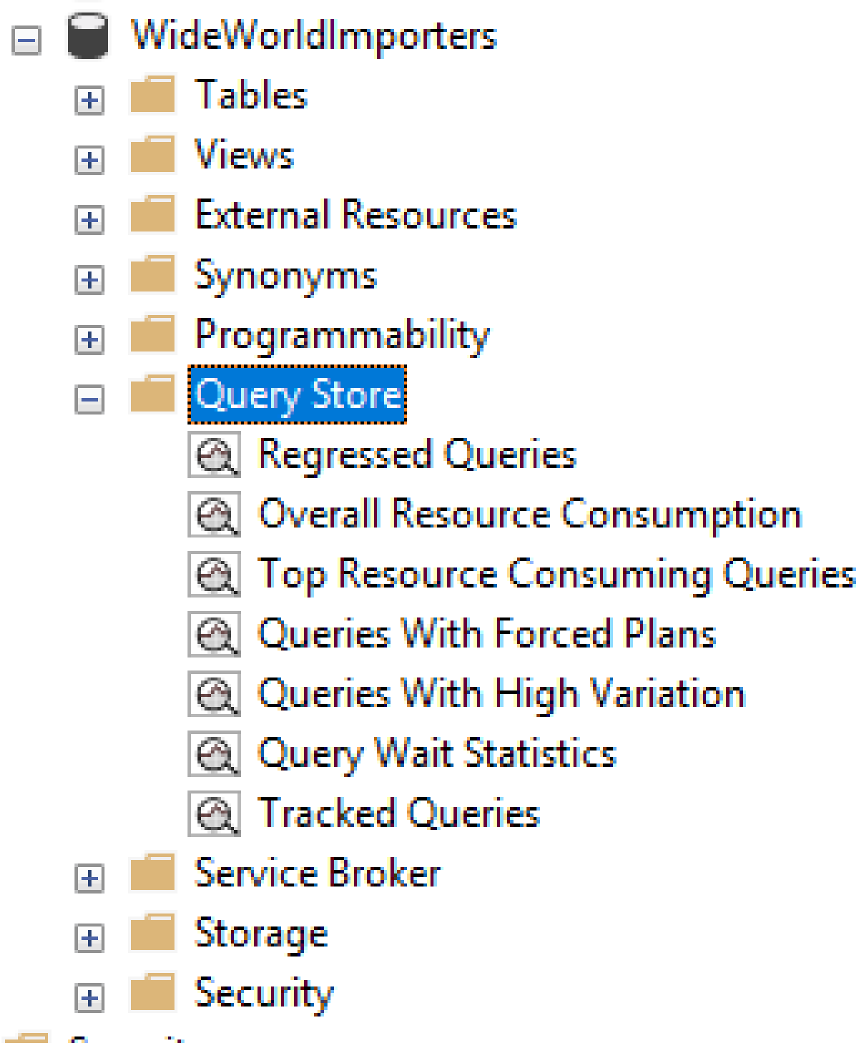
Database Properties - WideWorldImporters

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.

Current Disk Usage

WideWorldImporters	3,3 GB
Query Store Used	0,0 MB
Query Store Available	500,0 MB
Query Store Used	0,0 MB



CONFIGURING QUERY STORE

```
ALTER DATABASE WideWorldImporters SET QUERY_STORE = ON (
OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30),
DATA_FLUSH_INTERVAL_SECONDS = 900,
INTERVAL_LENGTH_MINUTES = 60,
MAX_STORAGE_SIZE_MB = 1000,
QUERY_CAPTURE_MODE = AUTO,
SIZE_BASED_CLEANUP_MODE = AUTO,
MAX_PLANS_PER_QUERY = 200,
WAIT_STATS_CAPTURE_MODE = ON,
)
```

CONFIGURING QUERY STORE

- OPERATION_MODE = { READ_WRITE | READ_ONLY | OFF | ERROR }
- CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30),
- DATA_FLUSH_INTERVAL_SECONDS = 900,
- INTERVAL_LENGTH_MINUTES = {1 | 5 | 10 | 15 | 30 | 60 | 1440}
- MAX_STORAGE_SIZE_MB = 1000,
- QUERY_CAPTURE_MODE = { ALL | AUTO | CUSTOM | NONE}
- SIZE_BASED_CLEANUP_MODE = { AUTO | OFF }
- MAX_PLANS_PER_QUERY = 200,
- WAIT_STATS_CAPTURE_MODE = { ON | OFF }

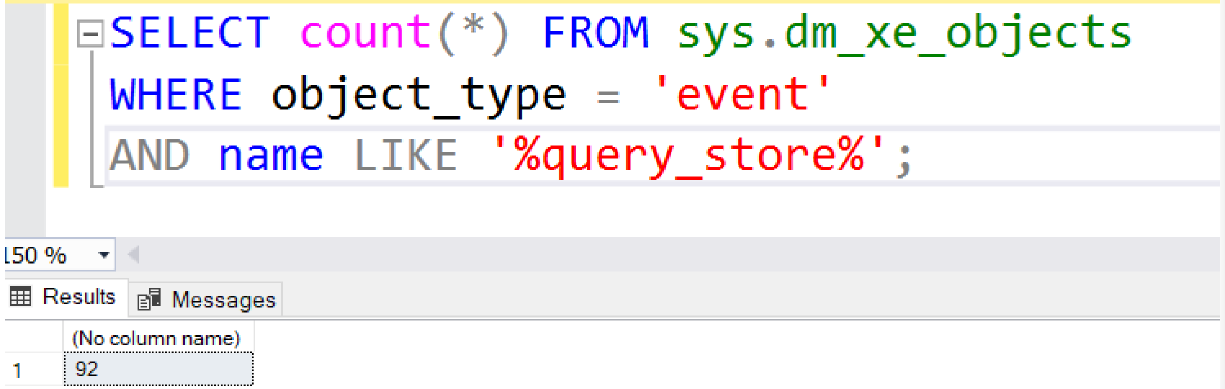
CONFIGURING QUERY STORE

```
ALTER DATABASE WideWorldImporters SET QUERY_STORE = ON (
QUERY_CAPTURE_MODE = CUSTOM,
     QUERY_CAPTURE_POLICY = (
     EXECUTION_COUNT = 30,
     TOTAL_COMPILE_CPU_TIME_MS = 1000,
     TOTAL_EXECUTION_CPU_TIME_MS = 100,
     STALE_CAPTURE_POLICY_THRESHOLD = 24 HOURS
    )
)
```

QUERY STORE IN ACTION

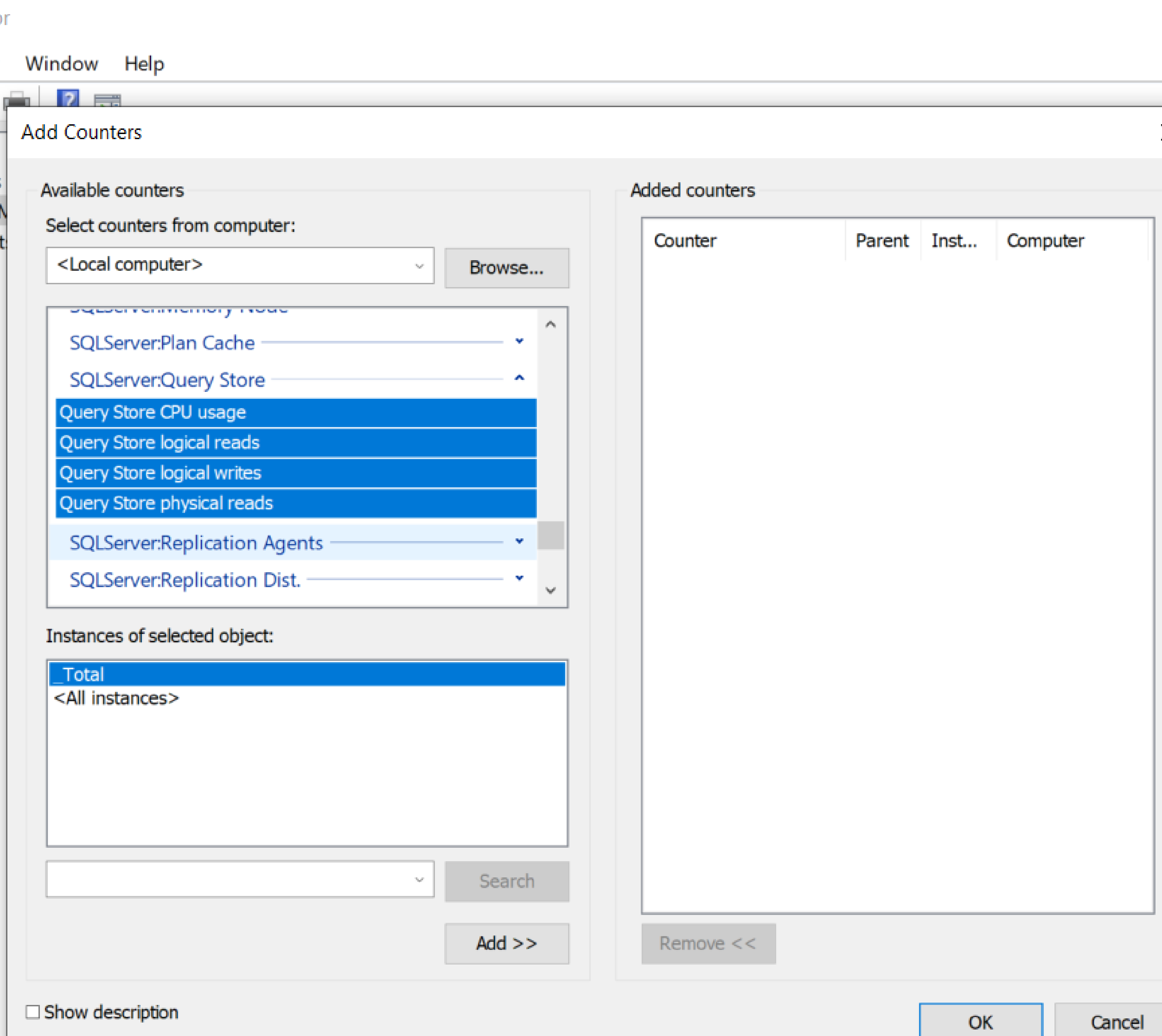
PERFORMANCE IMPACT

- 3-5%
- 4 new Query Store perf counters
- Extended Events:
  - SQL 2017 – 91 event
  - SQL 2019 – 92 events



```
SELECT count(*) FROM sys.dm_xe_objects
WHERE object_type = 'event'
AND name LIKE '%query_store%';
```

	(No column name)
1	92



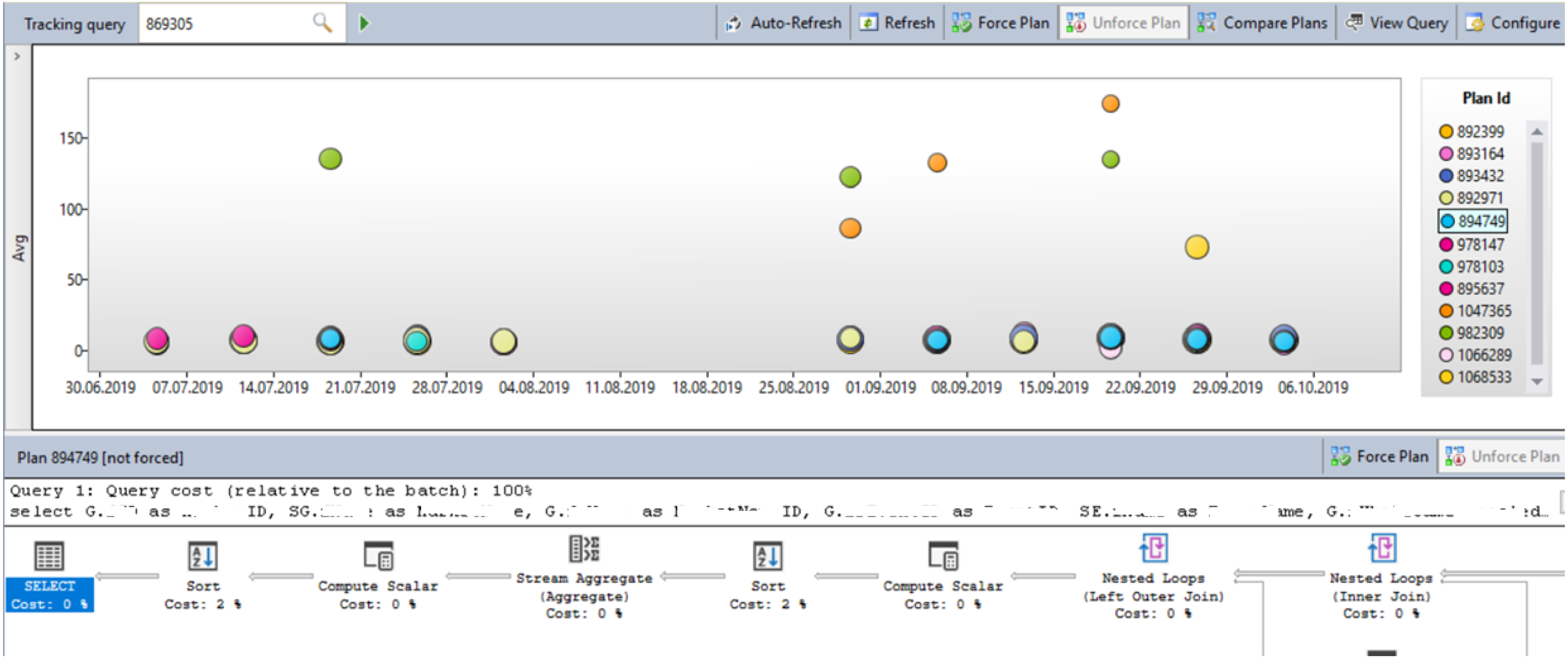
Query Store CPU usage
Query Store logical reads
Query Store logical writes
Query Store physical reads

_Total
<All instances>

PLAN FORCING

- Should be considered as mitigation and not a solution
  - Buying time to fix the query
  - Not always a good idea!
  - Check carefully plan history for a *query_id* before you force the plan
- Be careful with parameter sensitive queries
  - Plan forcing is NOT a solution for parameter sniffing!
- Use **sp_query_store_force_plan** rather than GUI
- Requires **db_owner** permissions
- You cannot force a plan for query if it hasn't been generated for that query
- You cannot force the plan if Query Store is in the *OFF* operation mode (in *Read-Only* is possible)

WHICH PLAN TO FORCE?



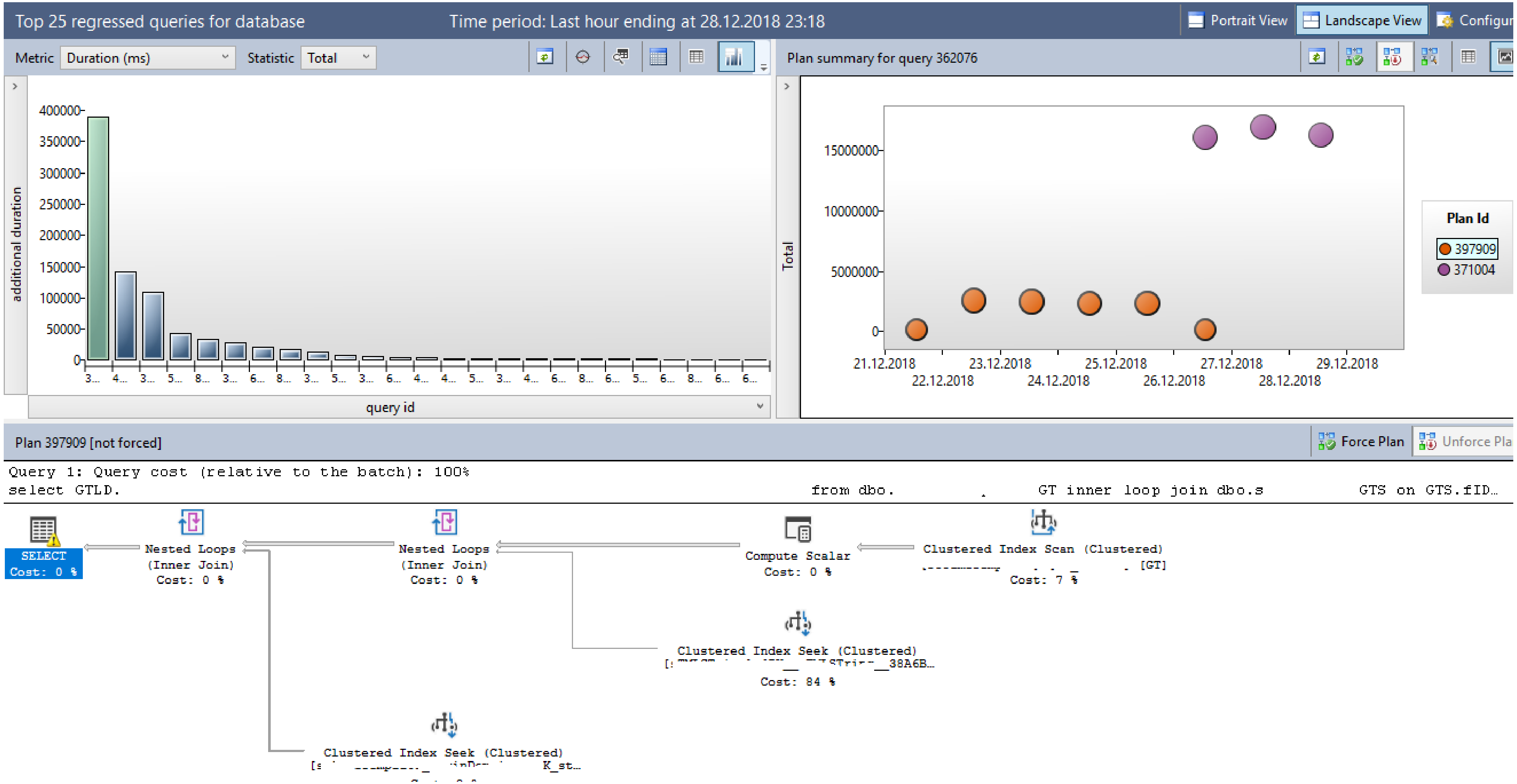
Plan 894749 [not forced]

```
Query 1: Query cost (relative to the batch): 100%
select G.[redacted] as [redacted] ID, SG.[redacted] as [redacted] e, G.[redacted] as [redacted] ID, G.[redacted] as [redacted] SE.[redacted] as [redacted] ame, G.[redacted] d…
```

QUERY STORE REPORTS

- Regressed Queries
- Overall Resource Consumption
- Top Resource Consuming Queries
- Queries With Forced Plans
- Queries With High Variation
- Query Waits Statistics
- Tracked Queries

REGRESSED QUERIES



Top 25 regressed queries for database

Time period: Last hour ending at 28.12.2018 23:18

Metric: Duration (ms)

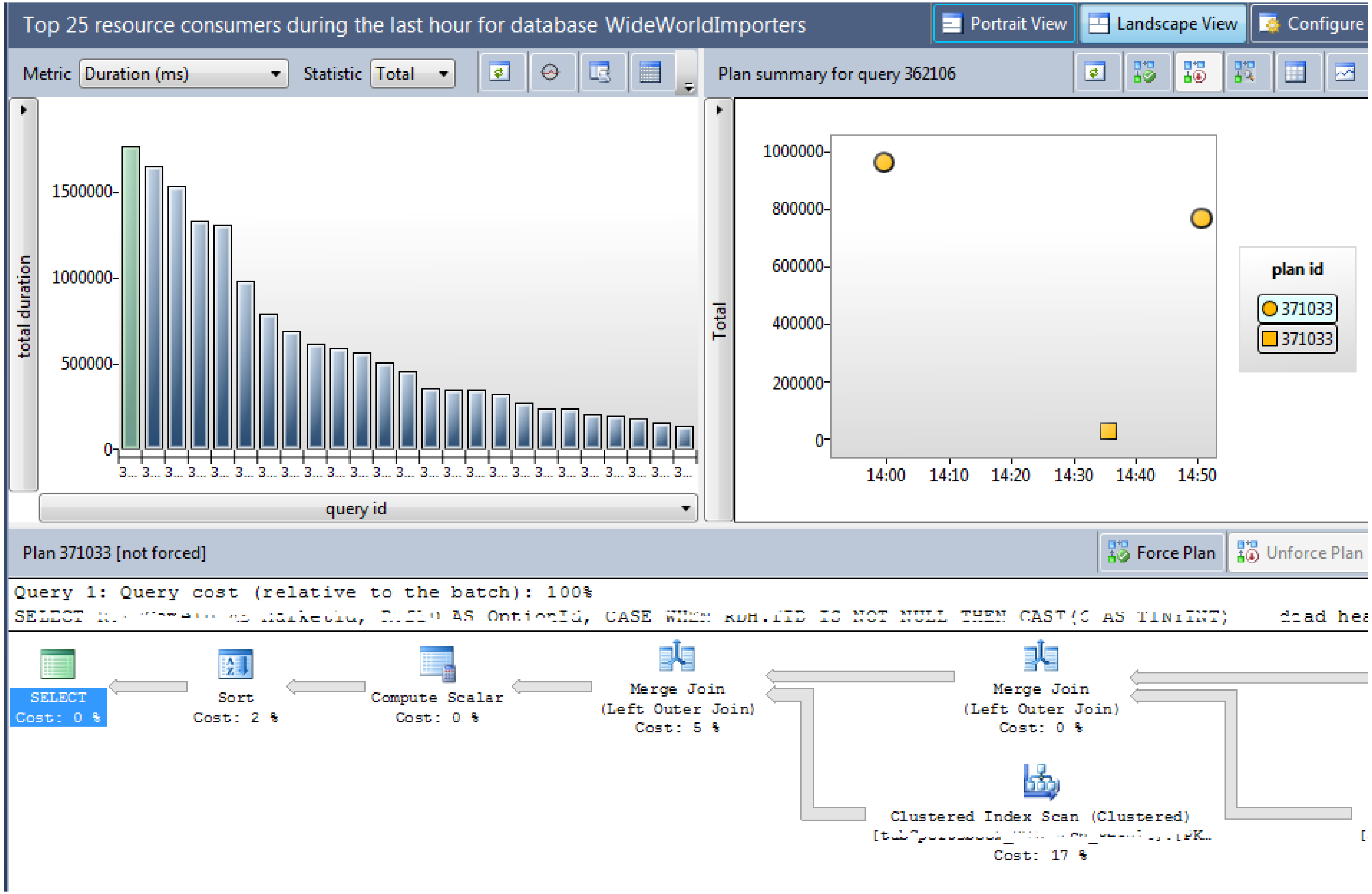
Statistic: Total

Plan summary for query 362076

Plan 397909 [not forced]

```
Query 1: Query cost (relative to the batch): 100%
select GTLD.                                    from dbo.          .    GT inner loop join dbo.s          GTS on GTS.fID…
```

TOP RESOURCE CONSUMING QUERIES



Top 25 resource consumers during the last hour for database WideWorldImporters

Metric: Duration (ms)

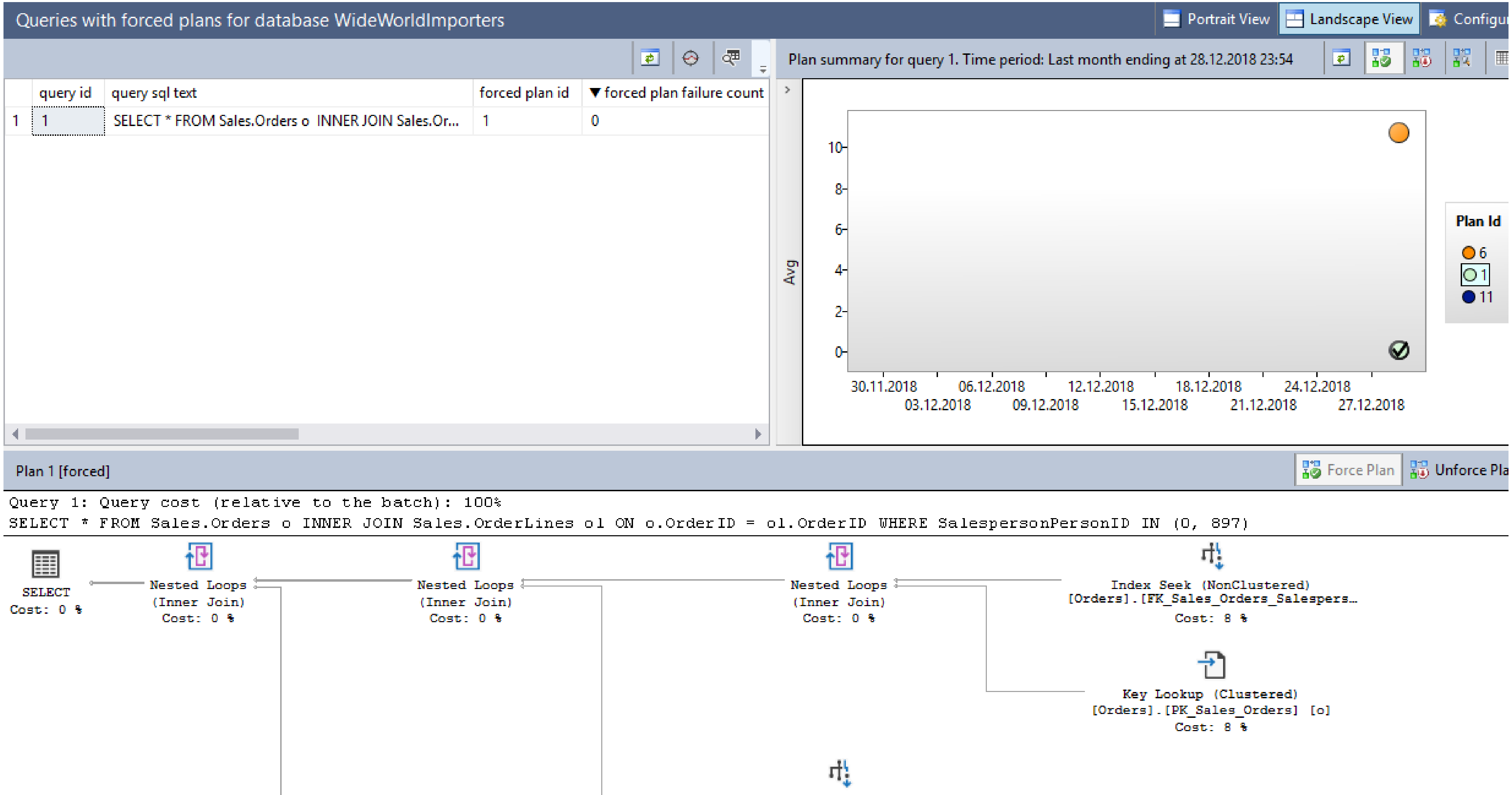
Statistic: Total

Plan summary for query 362106

Plan 371033 [not forced]

```
Query 1: Query cost (relative to the batch): 100%
SELECT [illegible] AS MarketId, [illegible] AS OptionId, CASE WHEN RDH.TID IS NOT NULL THEN CAST([illegible] AS TINYINT)     [illegible]
```

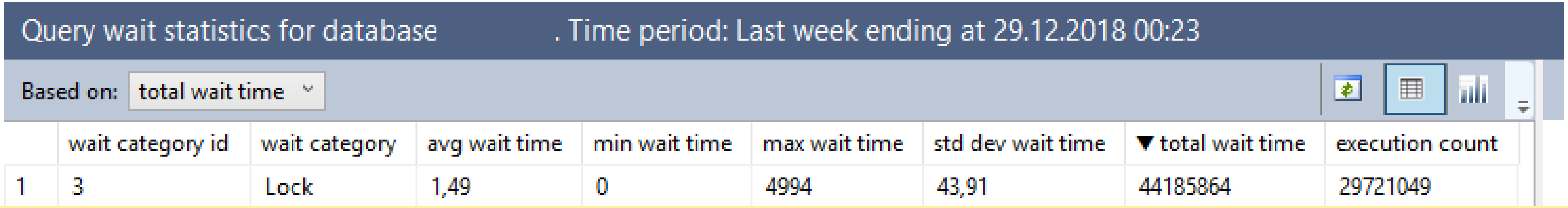
QUERIES WITH FORCED PLANS



	query id	query sql text	forced plan id	▼ forced plan failure count
1	1	SELECT * FROM Sales.Orders o INNER JOIN Sales.Or...	1	0

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Sales.Orders o INNER JOIN Sales.OrderLines ol ON o.OrderID = ol.OrderID WHERE SalespersonPersonID IN (0, 897)
```

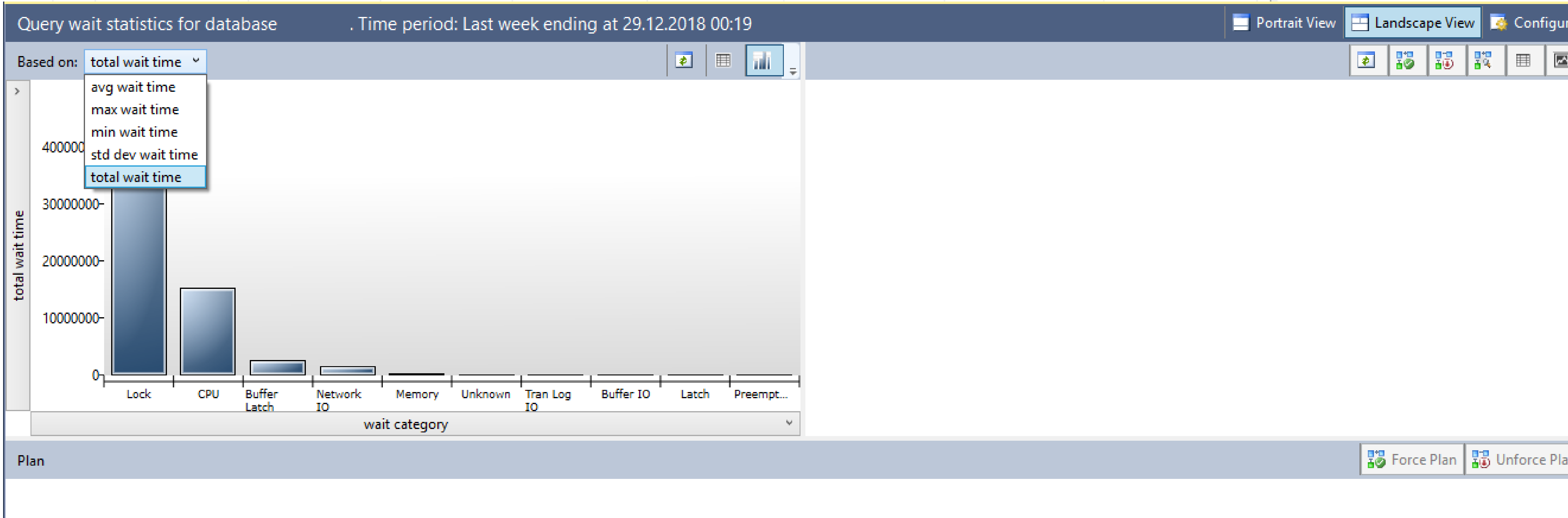
QUERY WAIT STATISTICS



Query wait statistics for database . Time period: Last week ending at 29.12.2018 00:23

Based on: total wait time

	wait category id	wait category	avg wait time	min wait time	max wait time	std dev wait time	▼ total wait time	execution count
1	3	Lock	1,49	0	4994	43,91	44185864	29721049



Query wait statistics for database . Time period: Last week ending at 29.12.2018 00:19

Based on: total wait time

- avg wait time
- max wait time
- min wait time
- std dev wait time
- total wait time

CAPTURING WAITS WITH QUERY STORE

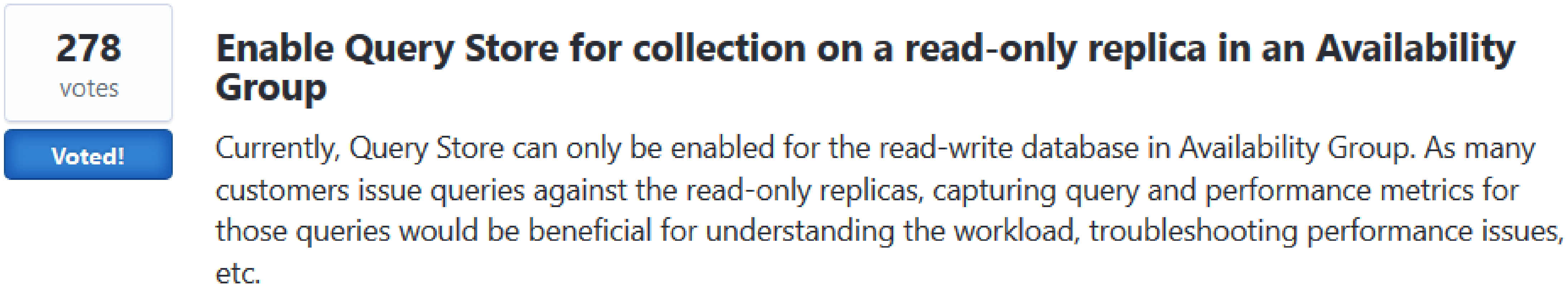
CAPTURING WAITS

Wait category	Wait types
CPU	SOS_SCHEDULER_YIELD
Memory	RESOURCE_SEMAPHORE, CMEMTHREAD, CMEMPARTITIONED, EE_PMOLOCK, MEMORY_ALLOCATION_EXT, RESERVED_MEMORY_ALLOCATION_EXT, MEMORY_GRANT_UPDATE
Network IO	ASYNC_NETWORK_IO, NET_WAITFOR_PACKET, PROXY_NETWORK_IO, EXTERNAL_SCRIPT_NETWORK_IOF
Parallelism	CXPACKET, EXCHANGE
Lock	LCK_M_%
Latch	LATCH_%

QUERY STORE LIMITATIONS

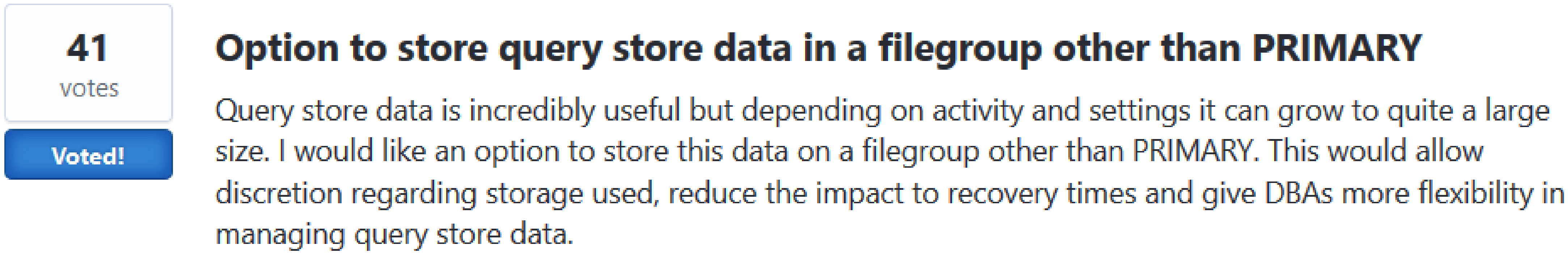
- Option to store query store data in a filegroup other than PRIMARY
- DDL statements are not captured
- Tracking queries, where we changed the query (tuned it)
- GUI - Object view (SP)
- Enable Query Store for collection on a read-only replica in an Availability Group
- Exceptions are not caught for Hekaton

MS AZURE FEEDBACK ITEMS



Enable Query Store for collection on a read-only replica in an Availability Group

Currently, Query Store can only be enabled for the read-write database in Availability Group. As many customers issue queries against the read-only replicas, capturing query and performance metrics for those queries would be beneficial for understanding the workload, troubleshooting performance issues, etc.



Option to store query store data in a filegroup other than PRIMARY

Query store data is incredibly useful but depending on activity and settings it can grow to quite a large size. I would like an option to store this data on a filegroup other than PRIMARY. This would allow discretion regarding storage used, reduce the impact to recovery times and give DBAs more flexibility in managing query store data.

QUERY STORE ISSUES – LOAD AND UNLOAD

- **Loading**
  - By default, all queries are blocked from running in the database until Query Store is loaded
  - TF 7752 Query Store load asynchronously, and in the background will be in a read-only state until it   is completely loaded, so queries can process while Query Store loads, but you will not be capturing them.
- **Shutdown**
  - shutdown is blocked by Query Store flushing process, you need to wait for recovery
  - TF 7745, you don’t wait, but data loss
- **ALTER DATABASE … SET QUERY_STORE = OFF**
  - Here TF 7745 does not work!

QUERY STORE - RECOMMENDATIONS

- Control the size, do not let it grow
  - faster, less prone to load/shutdown issues
  - if you need more data, move it to your own tables
- Choose a runtime interval which is proper for your workload
- Do not use ALL for QUERY_CAPTURE_MODE
- Use AUTO for SIZE_BASED_CLEANUP_MODE
- Overhead (too many adhoc queries, too many queries, small interval too many plans, more impact on large QS
- Do not use DROP/CREATE and db rename or dbid()

QUERY STORE - RECOMMENDATIONS

- DBAs/DevOps
  - Identify the problem fast
  - Very fast workaround implementation (w/o BL knowledge, w/o code changes)
  - Learning about potential issues
- Developers
  - Learning about workload and execution plans
  - Analyze workload patterns
  - It‘s not a definitive solution, but it‘s a great help