PASS

Performance Tuning with SQL Server 2017

Wien, 24.05.2018

Miloš Radivojević

Principal Database Consultant, bwin GVC

About me

Data Platform MVP

Principal Database Consultant, bwin GVC Vienna, Austria

Co-Founder: SQL Pass Austria

Conference Speaker

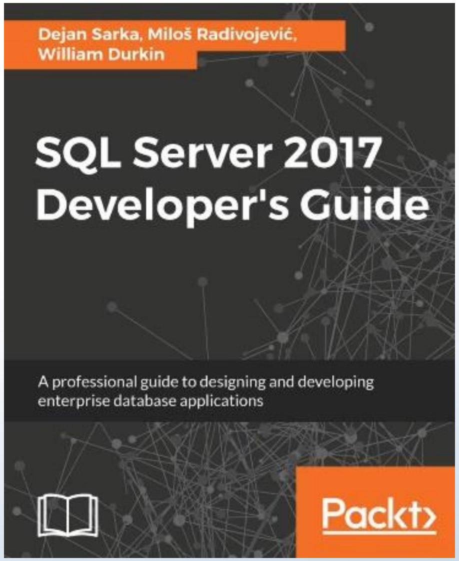
Book Co-Author

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>

Contact:
MRadivojevic@gvcgroup.com
Twitter: **@MilosSQL**

Agenda

New SQL Server (Microsoft) release cycles

Adaptive Query Processing

- Interleaved executions
- Batch Mode Adaptive Join
- Memory Grant feedback

Query Store as Game Changer

- Troubleshooting with Query Store
- Automatic tuning

New SQL Server (Microsoft) release cycles

Upgrade challenge (risk, not fully atomated test routines…)
Learning
Too frequent
Quality?
Abandoned services/features

Adaptive Query Processing

Adaptive Query Processing

Query Optimizer

- Chooses physical operators and creates the execution plan
- Estimates memory that is needed for query execution (Memory Grant)
- Based on estimates done by the Cardinality Estimator

Query Execution Issues

- Slow response time
- Intensive resource consuming
- Reduced throughput and concurrency

Adaptive Query Processing

SQL Server 2016 (and prior)

- After the execution plan is created, it is used in consecutive query executions, without changes (with the same operators and memory grants)

SQL Server 2017 Adaptive Query Processing

- Breaking the pipeline between query optimization and execution
- Executing a part of the query during the execution plan creation
- Updating a part of the cached plan during consecutive query executions (Memory Grant)
- Batch mode Adapter Join Operator

Adaptive Query Processing

Interleaved Execution

Batch Mode Memory Grant Feedback

Batch Mode Adaptive Join

Interleaved Execution

Related to queries with multi table valued functions (MTVF)

- Break the optimization process
- Execute the part of the query with function call and get actual cardinality
- Continue with the optimization process

Epilogue

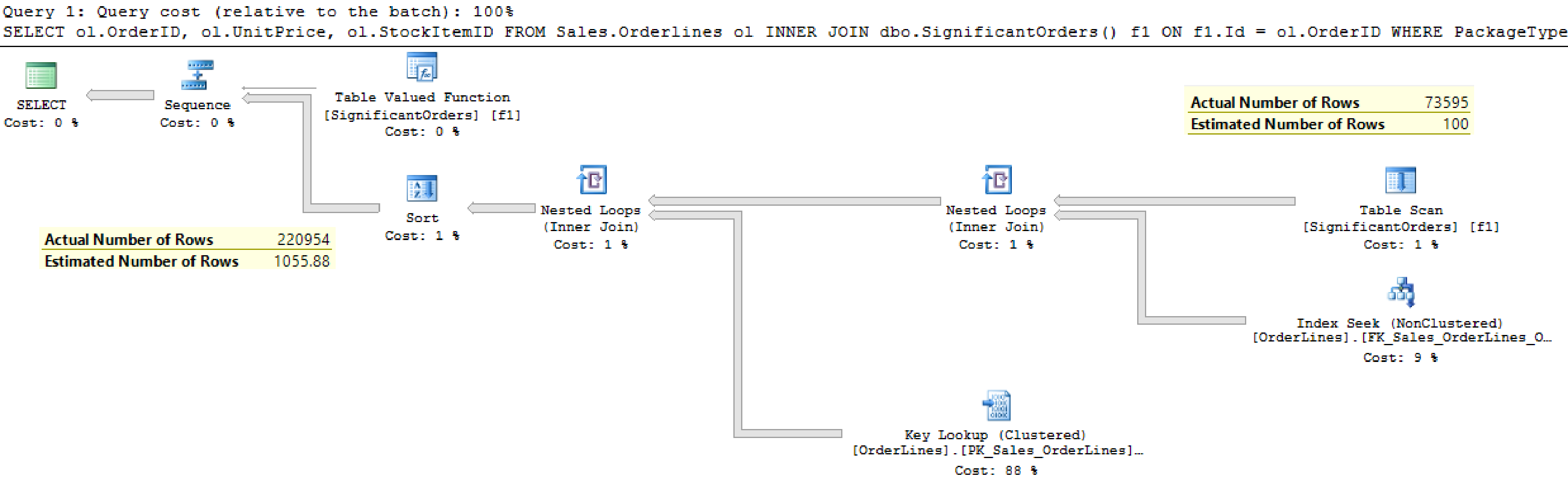
- More appropriate plan (correct cardinality instead of cardinality 100

Costs

- Increased CPU compile time
- Increased costs are acceptable, plan is usually better (sometimes significantly)

MTVF Execution

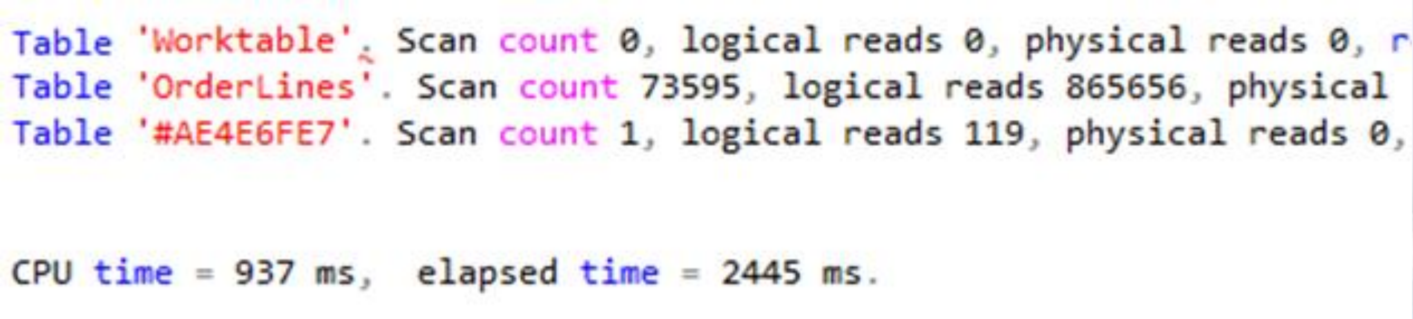
SQL Server 2016



```
Query 1: Query cost (relative to the batch): 100%
SELECT ol.OrderID, ol.UnitPrice, ol.StockItemID FROM Sales.Orderlines ol INNER JOIN dbo.SignificantOrders() f1 ON f1.Id = ol.OrderID WHERE PackageType
```

Actual Number of Rows	73595
Estimated Number of Rows	100

Actual Number of Rows	220954
Estimated Number of Rows	1055.88

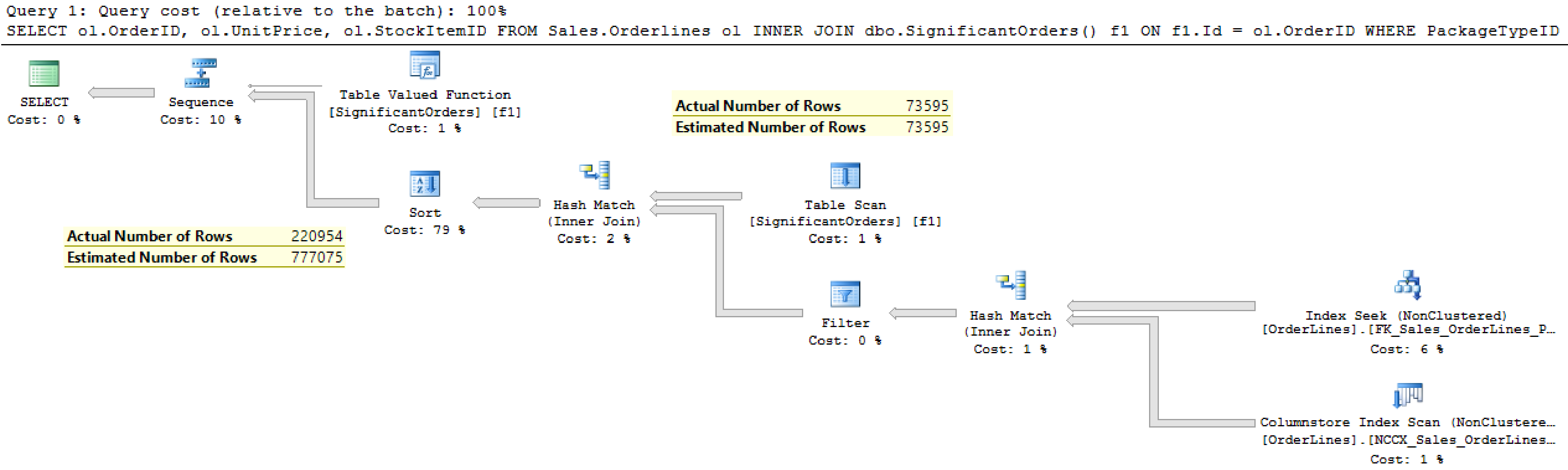


```
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, r
Table 'OrderLines'. Scan count 73595, logical reads 865656, physical
Table '#AE4E6FE7'. Scan count 1, logical reads 119, physical reads 0,


CPU time = 937 ms,  elapsed time = 2445 ms.
```

Interleaved Execution

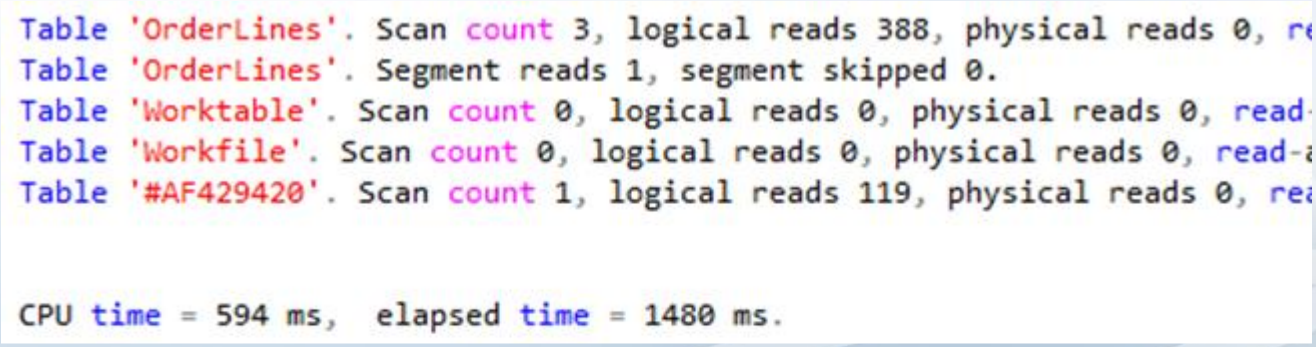
SQL Server 2017



```
Query 1: Query cost (relative to the batch): 100%
SELECT ol.OrderID, ol.UnitPrice, ol.StockItemID FROM Sales.Orderlines ol INNER JOIN dbo.SignificantOrders() f1 ON f1.Id = ol.OrderID WHERE PackageTypeID
```

Actual Number of Rows 73595
Estimated Number of Rows 73595

Actual Number of Rows 220954
Estimated Number of Rows 777075



```
Table 'OrderLines'. Scan count 3, logical reads 388, physical reads 0, re
Table 'OrderLines'. Segment reads 1, segment skipped 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-
Table '#AF429420'. Scan count 1, logical reads 119, physical reads 0, rea


CPU time = 594 ms,  elapsed time = 1480 ms.
```

Batch Mode Memory Grant Feedback

- Adjust memory grant parameter in the execution plan AFTER the plan is generated

- Monitors the execution of the query and if memory grant is constantly over- or underestimated, it recalculates and adjust it

- Requires a columnstore index on the affected table

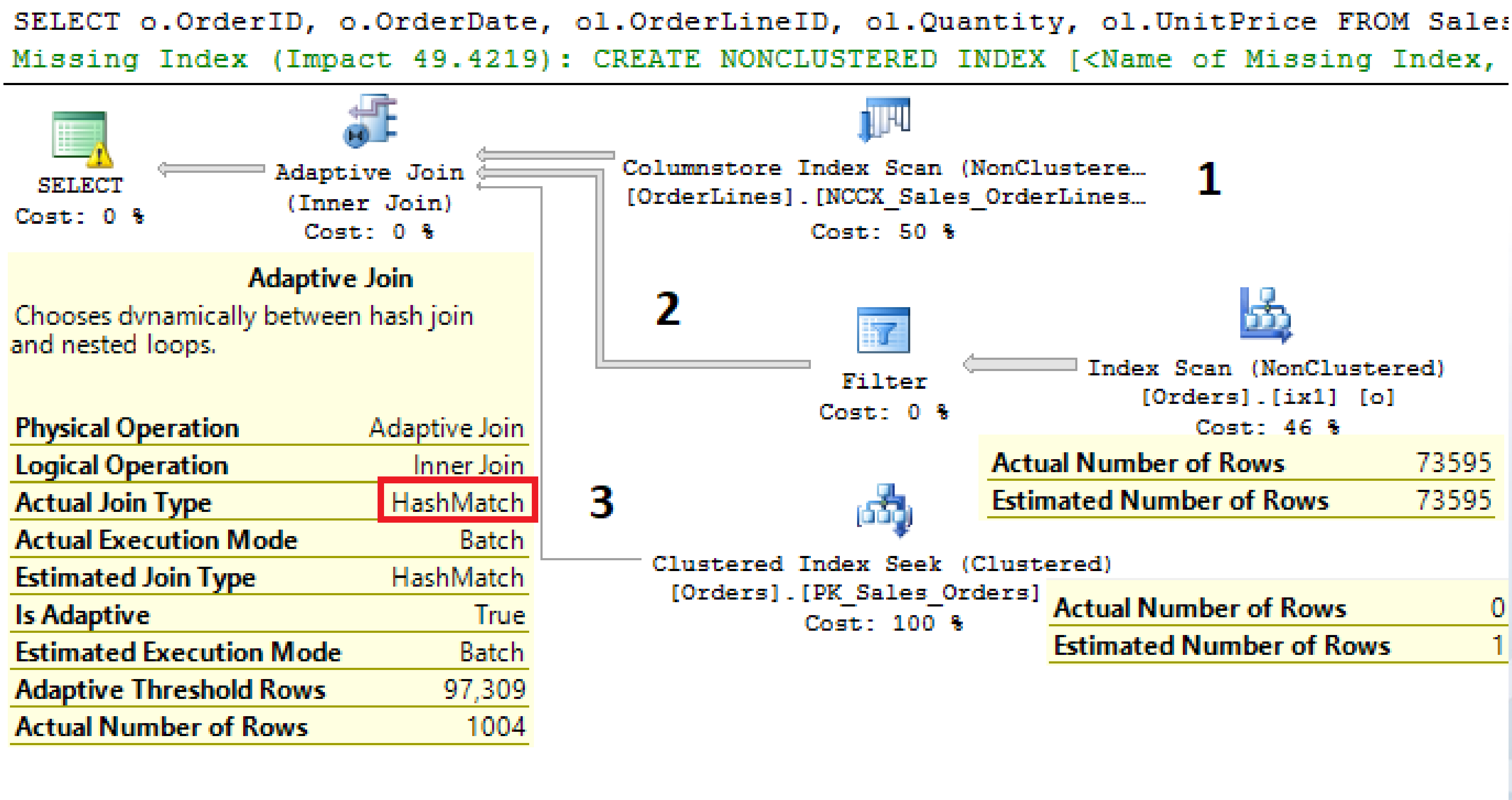
- If memory grant memory values oscillate, the feature is disabled

Batch Mode Adaptive Join

- New opeartor- Adaptive Join
- Allows to choose between Hash Join and Nested Loop Join operators after scanning an input
- Defines a threshold value that can be use for decision which operator to use
- It starts as Hash Join and if after input scanning estimated number of rows is less than threshold, it switches to Nested Loop Join
- Generaly, it will better handle some queries with variaty of parameters, but it will not solve all issues caused by wrongly chosen Join operator
- It can be disabled

Adaptive Join Operator

```
EXEC dbo.GetSomeOrderDeatils 112;
```



```
SELECT o.OrderID, o.OrderDate, ol.OrderLineID, ol.Quantity, ol.UnitPrice FROM Sales
Missing Index (Impact 49.4219): CREATE NONCLUSTERED INDEX [<Name of Missing Index,
```

SELECT
Cost: 0 %
Adaptive Join
(Inner Join)
Cost: 0 %
Columnstore Index Scan (NonClustere…
[OrderLines].[NCCX_Sales_OrderLines…
Cost: 50 %
1
2
Filter
Cost: 0 %
Index Scan (NonClustered)
[Orders].[ix1] [o]
Cost: 46 %
3
Clustered Index Seek (Clustered)
[Orders].[PK_Sales_Orders]
Cost: 100 %

Adaptive Join

Chooses dvnamically between hash join and nested loops.

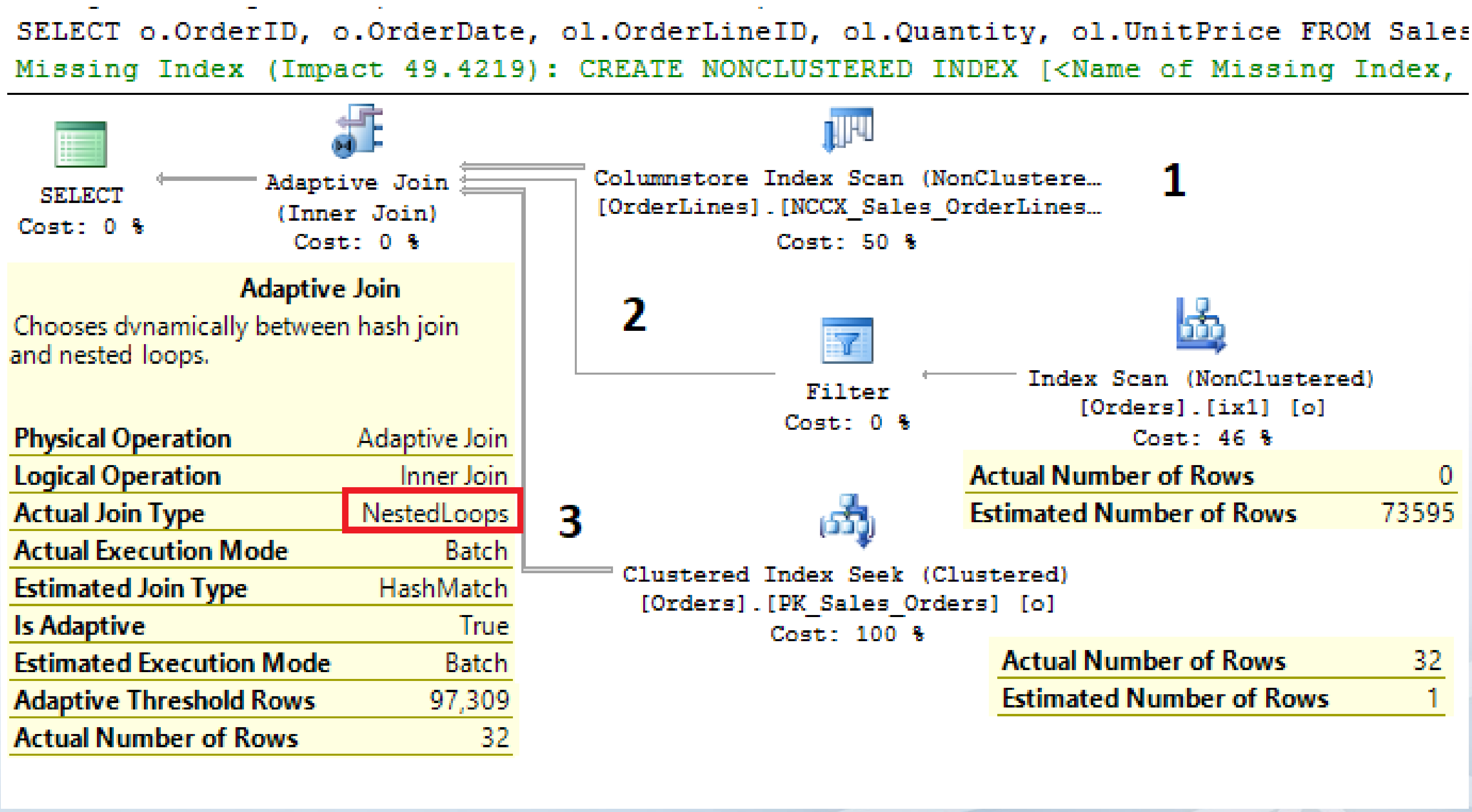
Physical Operation	Adaptive Join
Logical Operation	Inner Join
Actual Join Type	HashMatch
Actual Execution Mode	Batch
Estimated Join Type	HashMatch
Is Adaptive	True
Estimated Execution Mode	Batch
Adaptive Threshold Rows	97,309
Actual Number of Rows	1004

Actual Number of Rows	73595
Estimated Number of Rows	73595

Actual Number of Rows	0
Estimated Number of Rows	1

Adaptive Join Operator

```
EXEC dbo.GetSomeOrderDeatils 1;
```



```
SELECT o.OrderID, o.OrderDate, ol.OrderLineID, ol.Quantity, ol.UnitPrice FROM Sales
Missing Index (Impact 49.4219): CREATE NONCLUSTERED INDEX [<Name of Missing Index,
```

SELECT
Cost: 0 %
Adaptive Join
(Inner Join)
Cost: 0 %
Columnstore Index Scan (NonClustere…
[OrderLines].[NCCX_Sales_OrderLines…
Cost: 50 %
1
2
Filter
Cost: 0 %
Index Scan (NonClustered)
[Orders].[ix1] [o]
Cost: 46 %
3
Clustered Index Seek (Clustered)
[Orders].[PK_Sales_Orders] [o]
Cost: 100 %

Adaptive Join

Chooses dvnamically between hash join and nested loops.

Physical Operation	Adaptive Join
Logical Operation	Inner Join
Actual Join Type	NestedLoops
Actual Execution Mode	Batch
Estimated Join Type	HashMatch
Is Adaptive	True
Estimated Execution Mode	Batch
Adaptive Threshold Rows	97,309
Actual Number of Rows	32

Actual Number of Rows	0
Estimated Number of Rows	73595

Actual Number of Rows	32
Estimated Number of Rows	1

Adaptive Join Operator

Adaptive Join Operator brings overhead

How to disable it:

- OPTION(USE HINT('DISABLE_BATCH_MODE_ADAPTIVE_JOINS'));
- ALTER DATABASE SCOPED CONFIGURATION SET DISABLE_BATCH_MODE_ADAPTIVE_JOINS = ON;

Query Store and Automatic Tuning

Query Store in SQL Server 2016

New troubleshooting tool

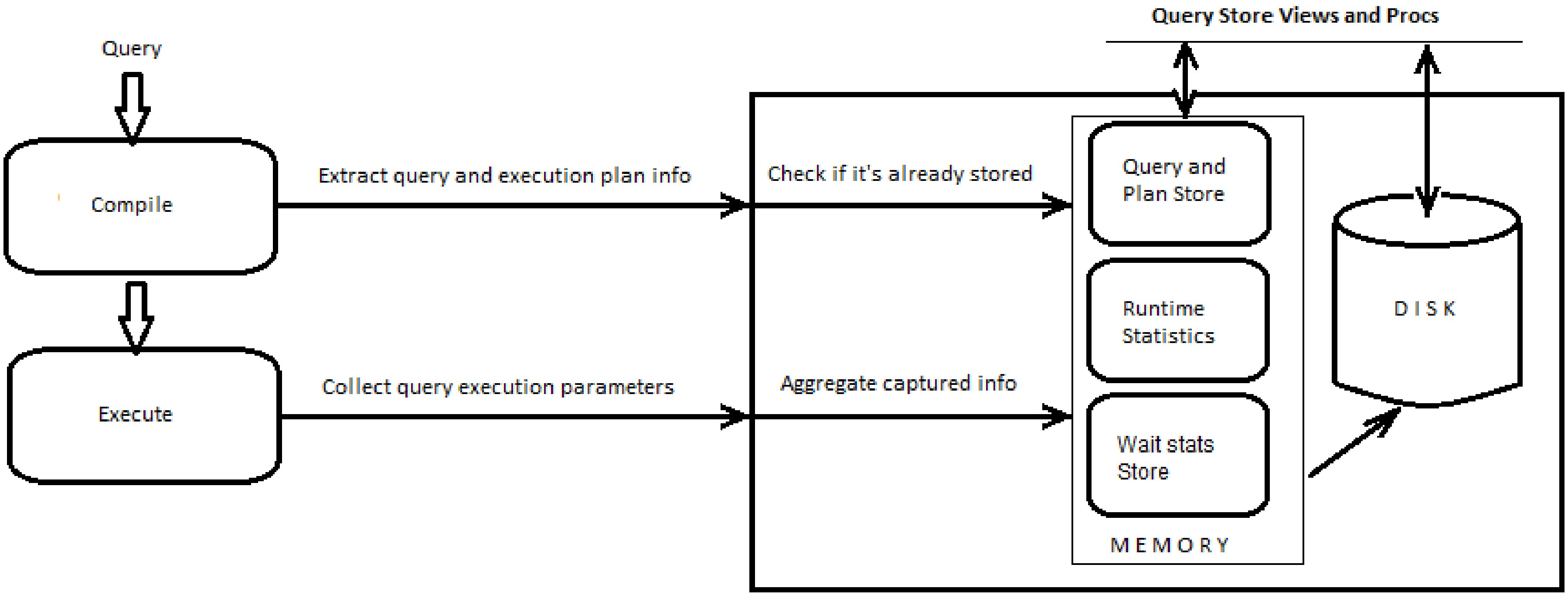
- Captures all execution relevant parameters for database queries
- Information are persistent, belongs to the database
- Quick identify performance regressions
- Helps you to learn how your database workload changes over time
- Helps you to identify queries that did not execute successfully
- Allows you to fix some performance issues

Query Store in SQL Server 2017

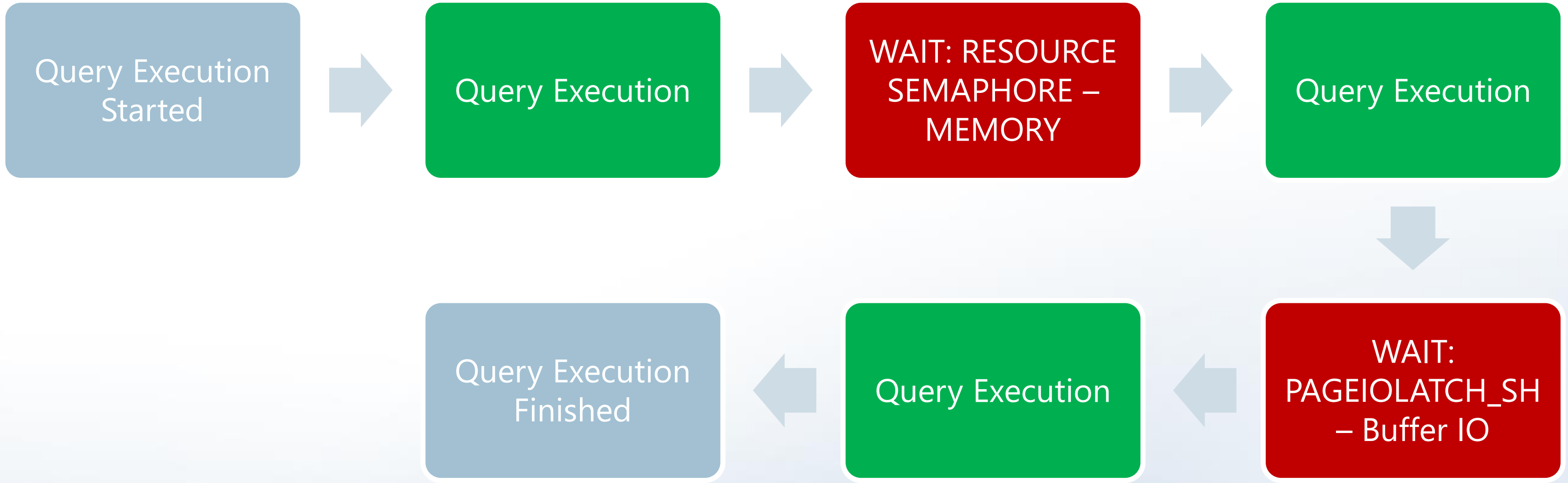
New features and enhancements

- Query Store captures wait stats (24 wait stats categories)
- Tuning Recommendations
- Automatic Tuning

Query Store in SQL Server 2017


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 in SQL Server 2017


Query Execution Started

Query Execution

WAIT: RESOURCE SEMAPHORE – MEMORY

Query Execution

WAIT: PAGEIOLATCH_SH – Buffer IO

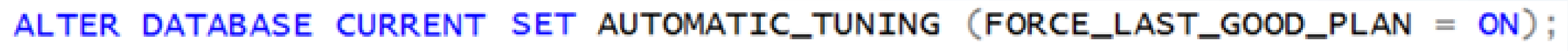
Query Execution

Query Execution Finished

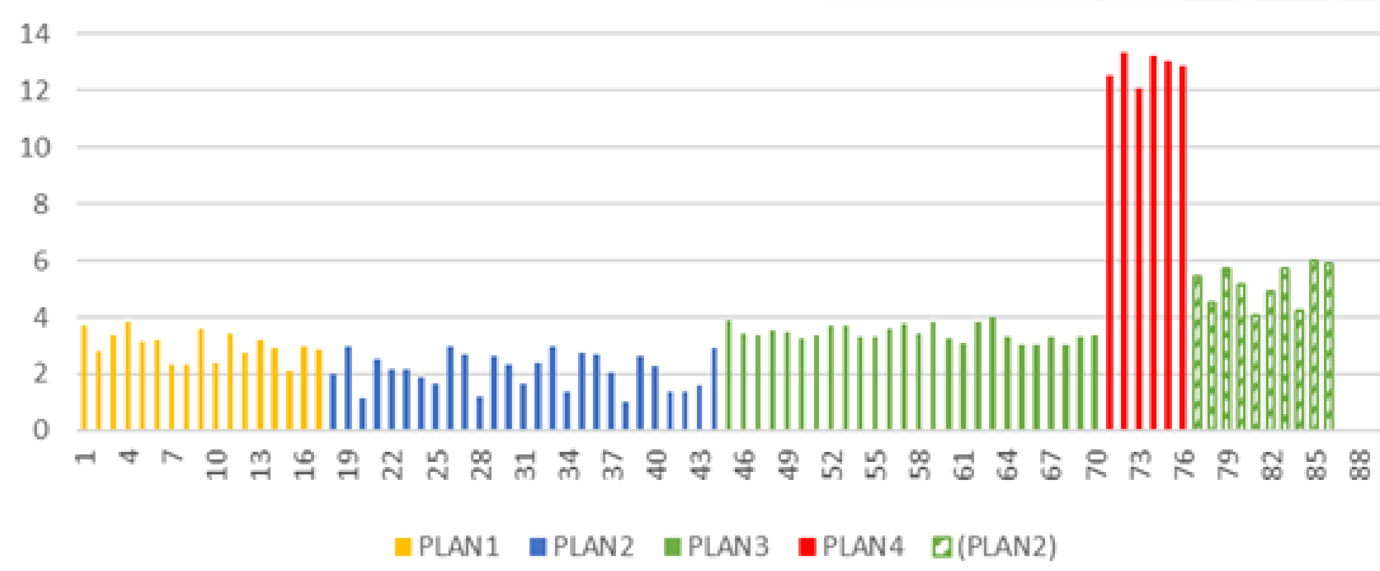
SQL Server 2017 Automatic Tuning

Two options:

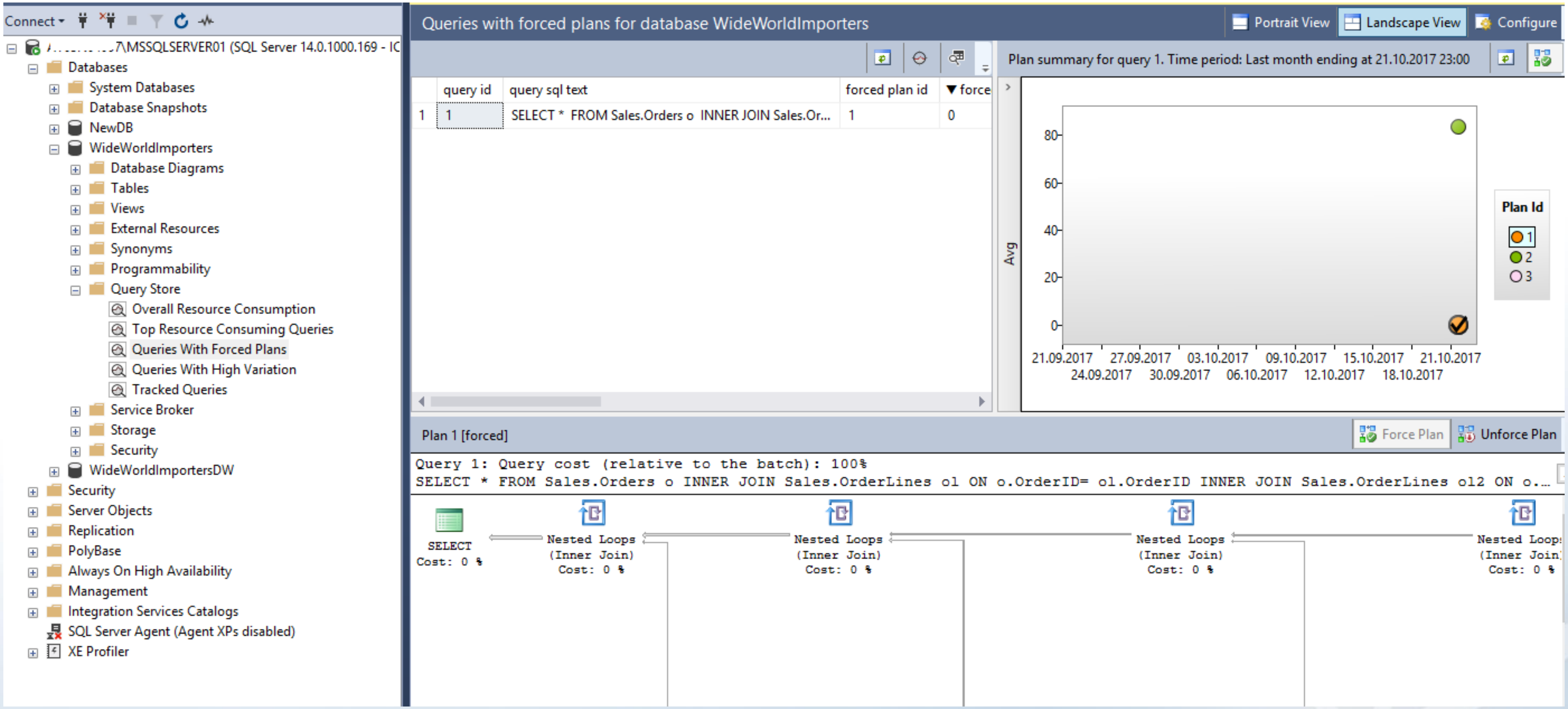
- Offline: Recommended actions via DMV **sys.dm_db_tuning_recommendations**
- Online: automatically switch to the last known good plan whenever the regression is detected



```
ALTER DATABASE CURRENT SET AUTOMATIC_TUNING (FORCE_LAST_GOOD_PLAN = ON);
```



SQL Server 2017 Automatic Tuning



Connect ▾

- [illegible]7\MSSQLSERVER01 (SQL Server 14.0.1000.169 - IC
  - Databases
    - System Databases
    - Database Snapshots
    - NewDB
    - WideWorldImporters
      - Database Diagrams
      - Tables
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Query Store
        - Overall Resource Consumption
        - Top Resource Consuming Queries
        - Queries With Forced Plans
        - Queries With High Variation
        - Tracked Queries
      - Service Broker
      - Storage
      - Security
    - WideWorldImportersDW
  - Security
  - Server Objects
  - Replication
  - PolyBase
  - Always On High Availability
  - Management
  - Integration Services Catalogs
  - SQL Server Agent (Agent XPs disabled)
  - XE Profiler

Queries with forced plans for database WideWorldImporters

Portrait View | Landscape View | Configure

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

Plan summary for query 1. Time period: Last month ending at 21.10.2017 23:00

Plan 1 [forced]

Force Plan | Unforce Plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Sales.Orders o INNER JOIN Sales.OrderLines ol ON o.OrderID= ol.OrderID INNER JOIN Sales.OrderLines ol2 ON o...
```

PASS

Danke!

Microsoft

redgate
ingeniously simple tools

O'REILLY®