Microsoft

SQL Server Query Execution and Plans

Module 6

Learning Units covered in this Module

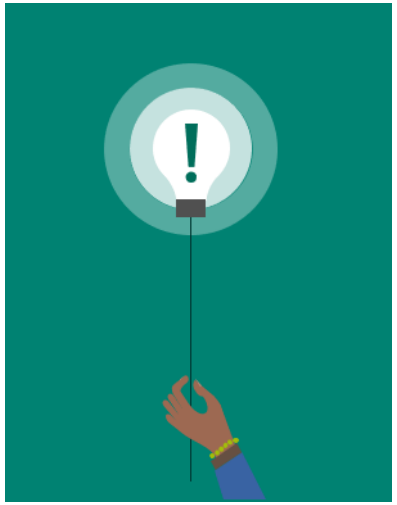
- Lesson 1: SQL Query Execution & Optimization
- Lesson 2: SQL Query Plan Analysis
- Lesson 3: SQL Server Query Store

Lesson 1: SQL Query Execution & Optimization

Objectives

After completing this learning, you will be able to:

- Explain Query Compilation and Optimization Process.
- Explain Query Execution Process.
- Explain Recompilation causes.



SQL Server Execution Plan

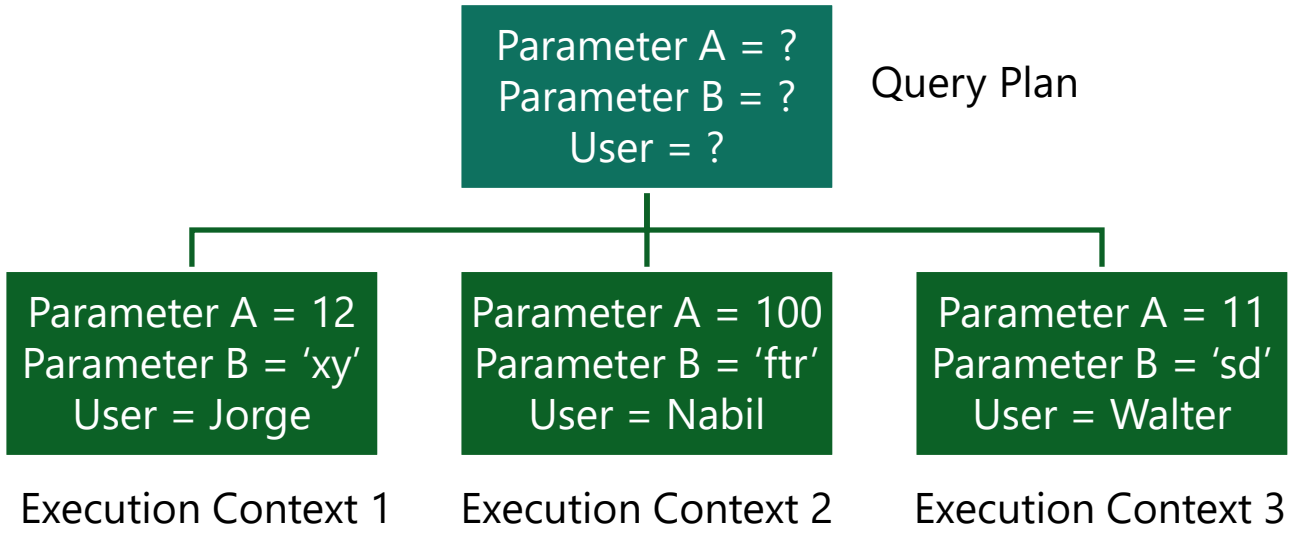
Main components

Compiled Plan (or Query Plan)

Compilation produces a query plan, which is a read-only data structure used by any number of users.

Execution Context

A data structure used to hold information specific to a query execution, such as parameter values.



SQL Server Execution Plan Caching

Overview

Part of the memory pool used to store execution plans – also known as plan cache.

The plan cache has two stores for all compiled plans:

The **Object Plans** cache store (OBJCP)

used for plans related to persisted objects (stored procedures, functions, and triggers).

The **SQL Plans** cache store (SQLCP)

used for plans related to autoparameterized, dynamic, or prepared queries.

SQL Server compilation and execution

Concepts

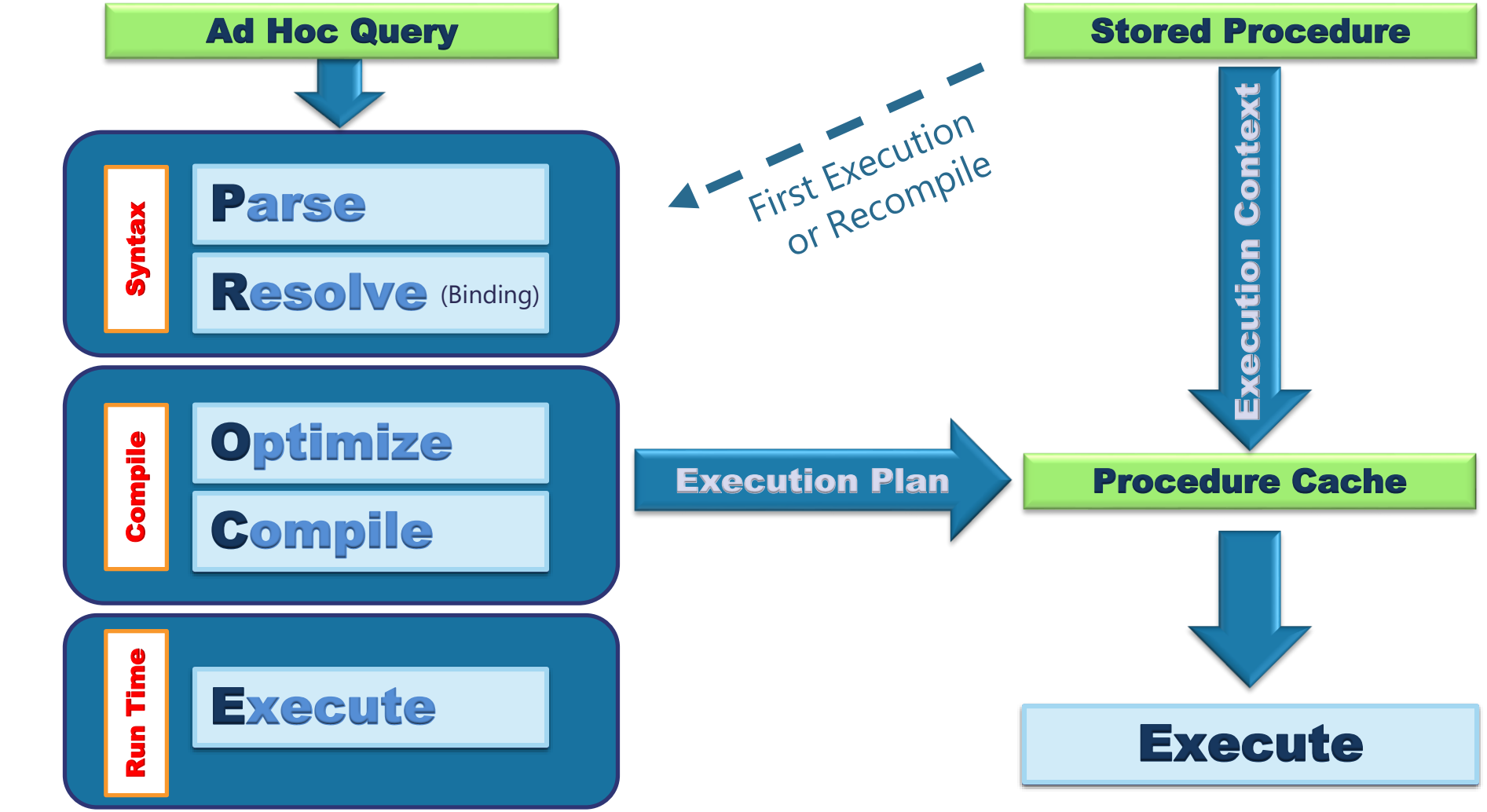
Compilation

Process of creating a good enough query execution plan, as quickly as possible for a query batch.

Refer to both the compilation of non-DML constructs in SQL statements (control flow, DDL, etc.) and the process of Query Optimization.

Query Execution

Process of executing the plan that is created during query compilation and optimization.



SQL

Sets

empid	lastname	firstna...	title	titleofcourt...	birthdate
1	Davis	Sara	CEO	Ms.	1958-12-08 00:00:00.000
2	Funk	Don	Vice President, Sales	Dr.	1962-02-19 00:00:00.000
3	Lew	Judy	Sales Manager	Ms.	1973-08-30 00:00:00.000
4	Peled	Yael	Sales Representative	Mrs.	1947-09-19 00:00:00.000
5	Buck	Sven	Sales Manager	Mr.	1965-03-04 00:00:00.000

What does the binding step resolve?

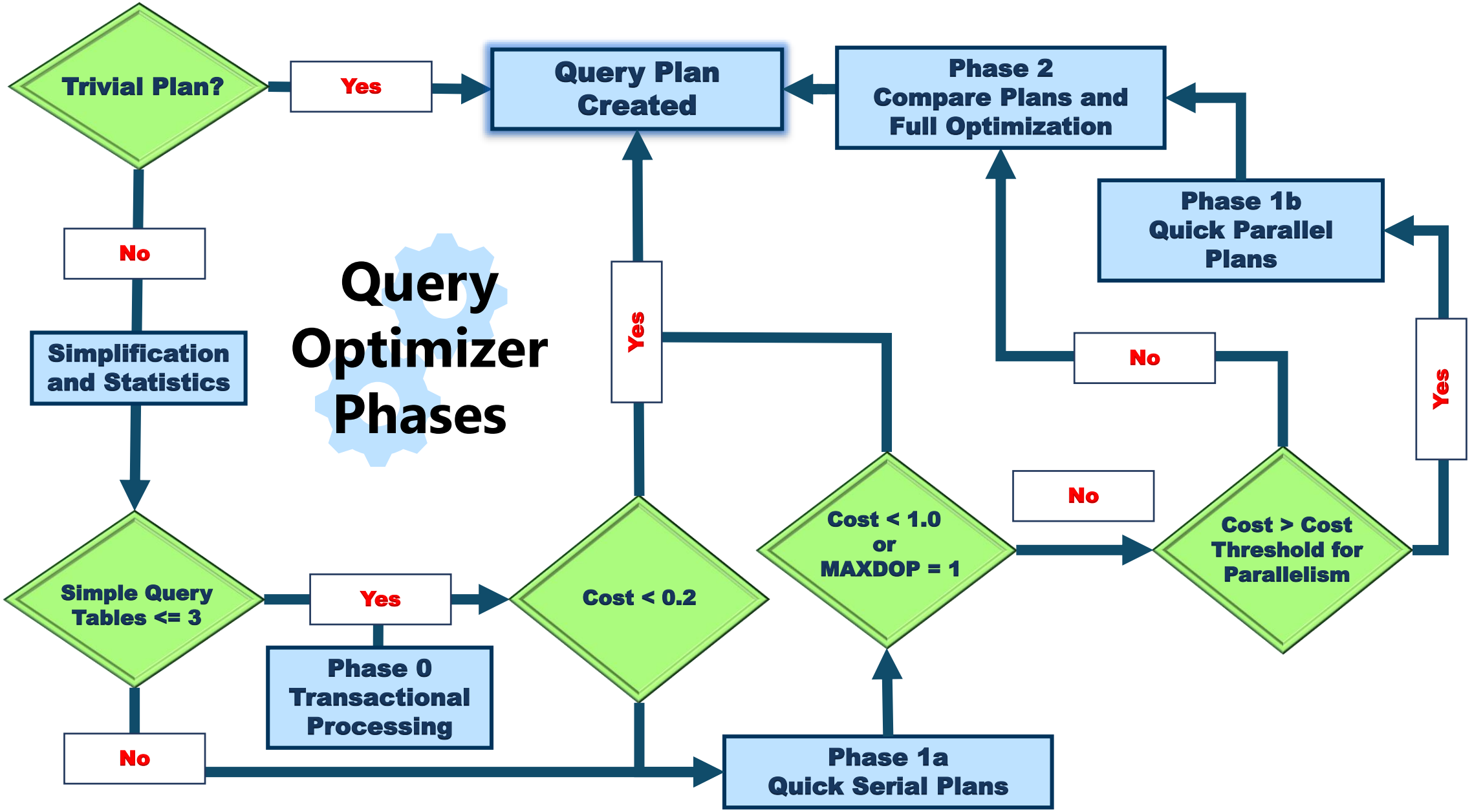
User permissions are checked.

Does a cached plan exist?

Object names (Tables, Views, Columns, etc.) to see if they exist.

Resolve aliases of columns and tables

Data types and if implicit data type conversions are needed.


Query Optimizer Phases
Trivial Plan?
Yes
Query Plan Created
Phase 2 Compare Plans and Full Optimization
Phase 1b Quick Parallel Plans
No
Simplification and Statistics
Yes
No
Yes
Simple Query Tables <= 3
Yes
Phase 0 Transactional Processing
Cost < 0.2
Cost < 1.0 or MAXDOP = 1
No
Cost > Cost Threshold for Parallelism
No
Phase 1a Quick Serial Plans

Query Simplification Process

Constant Folding: Expressions with constant values are reduced

- **Quantity = 2 + 3** becomes **Quantity = 5**
- **10 < 20** becomes **True**

Contradiction Detection: Removes criteria that doesn't match table constraints

- **Constraint:** Age > 18
- **Contradiction:** WHERE Age < 18

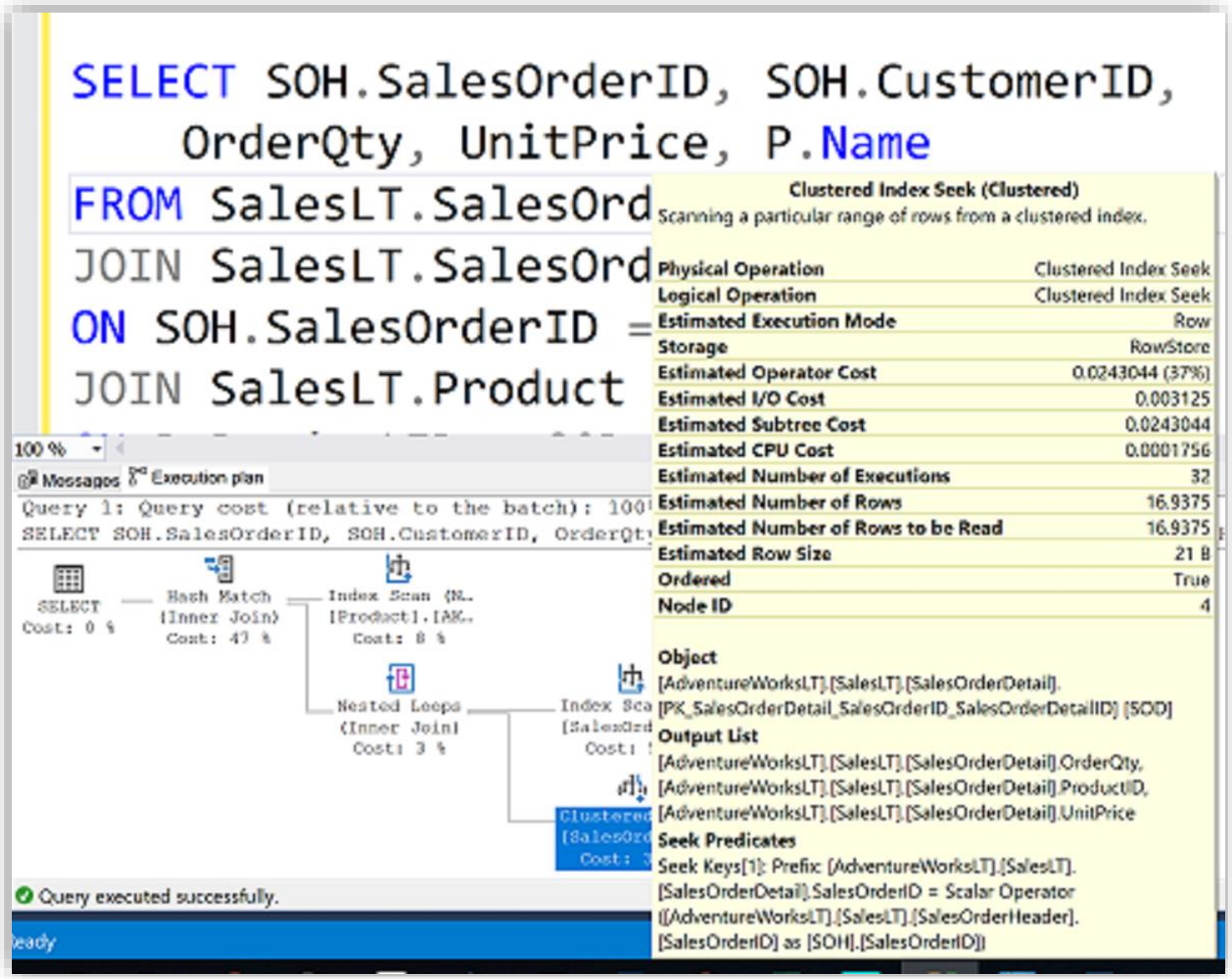
Domain Simplification: Reduces complex ranges to simple ranges

- **Complex range:** ID > 10 and ID < 20 or ID > 30 and < 50
- **Simplified range:** ID > 10 and < 50

Join Simplification: Removes redundant joins that are not necessary

Predicate Pushdown: Perform calculations only on rows returned

What is an Execution Plan?



How to see the query plan

Graphical execution plan

Estimated Execution Plan (Before Execution)

- The compiled plan.

Actual Execution Plan (After Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information available after the execution completes, such as execution warnings, or in newer versions of the Database Engine, the elapsed and CPU time used during execution.

Live Query Statistics (During Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information during execution progress and is updated every second. Runtime information includes for example the actual number of rows flowing through the operators.
- Enables rapid identification of potential bottlenecks.

SQL Server Execution Plan Recompilations

Overview

Most recompilations are required either for statement correctness or to obtain potentially faster query execution plan.

The engine detects changes that invalidate execution plan(s) and marks those as not valid. New plan must be recompiled for the next query execution.

Starting with SQL Server 2005, whenever a statement within a batch causes recompilation, only the statement inside the batch that triggers recompilation is recompiled.

SQL Server Execution Plan Recompilations

Recompilation reasons

Table / Index Changes

- Changes made to objects referenced by the query (ALTER TABLE and ALTER VIEW).
- Changing or dropping any indexes used by the execution plan.

Stored Procedures

- Changes made to a single procedure, which would drop all plans for that procedure from the cache (ALTER PROCEDURE).
- Explicit call to sp_recompile.
- Executing a stored procedure using the WITH RECOMPILE option.

Data Volume

- Updates on statistics used by the execution plan
- For tables with triggers, if the number of rows in the inserted or deleted tables grows significantly.

Other

- Large numbers of changes to keys (generated by statements from other users that modify a table referenced by the query).
- Temporary table changes

Questions?

Knowledge Check

What is meant by SQL Server's query optimizer being **cost-based**?

When is a query considered for a parallel execution plan?

Will SQL Server evaluate **every** possible query plan in the process of optimization? Why?

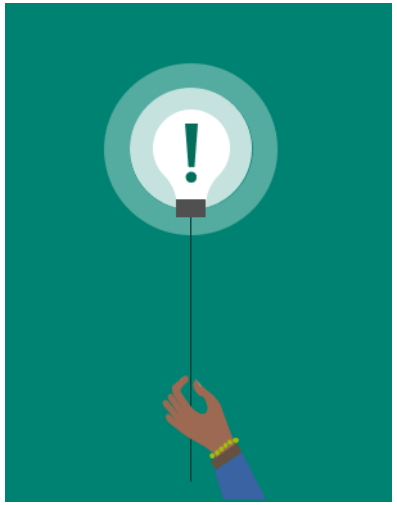
Name two recompilation causes.

Lesson 2: SQL Server Query Plan Analysis

Objectives

After completing this learning, you will be able to:

- Read execution plans.
- Understand logical and physical join operators.
- Describe data access.



How to see the query plan

Graphical execution plan

Estimated Execution Plan (Before Execution)

- The compiled plan.

Actual Execution Plan (After Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information available after the execution completes, such as execution warnings, or in newer versions of the Database Engine, the elapsed and CPU time used during execution.

Live Query Statistics (During Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information during execution progress and is updated every second. Runtime information includes for example the actual number of rows flowing through the operators.
- Enables rapid identification of potential bottlenecks.

Contents of an Execution Plan

- Sequence in which the source tables are accessed.
- Methods used to extract data from each table.
- How data is joined
- Use of temporary worktables and sorts
- Estimated rowcount, iterations, and costs from each operator
- Actual rowcount and iterations

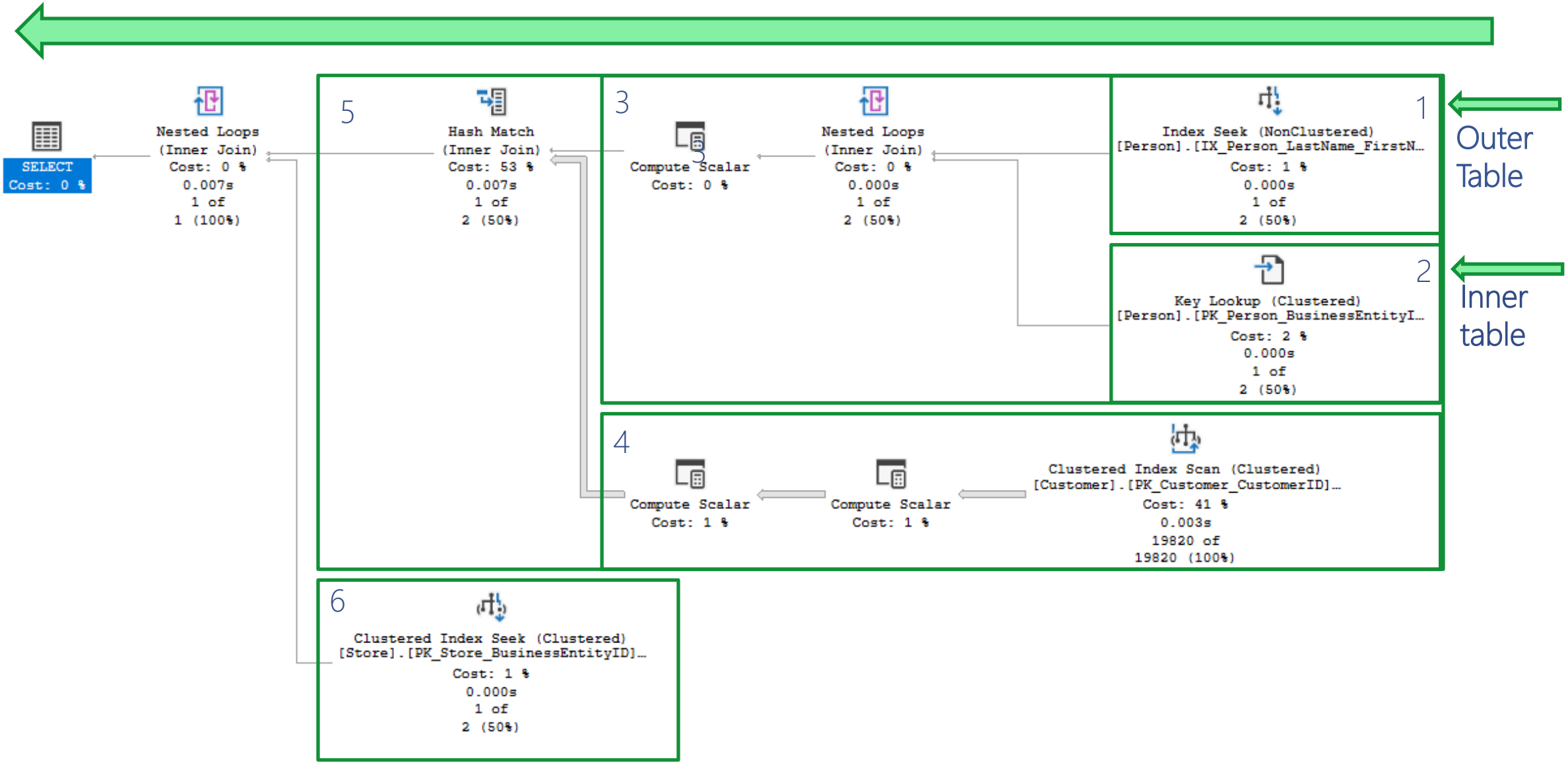
How to see the query plan

Text and XML

	Command	Execute query?	Include estimated row counts & stats (Estimated Query Plan)	Include actual row counts & stats (Actual Query Plan)
Text Plan	SET SHOWPLAN_TEXT ON	No	No	No
	SET SHOWPLAN_ALL ON	No	Yes	No
	SET STATISTICS PROFILE ON	Yes	Yes	Yes
XML Plan	SET SHOWPLAN_XML ON	No	Yes	No
	SET STATISTICS PROFILE XML	Yes	Yes	Yes

SSMS Graphical Plan

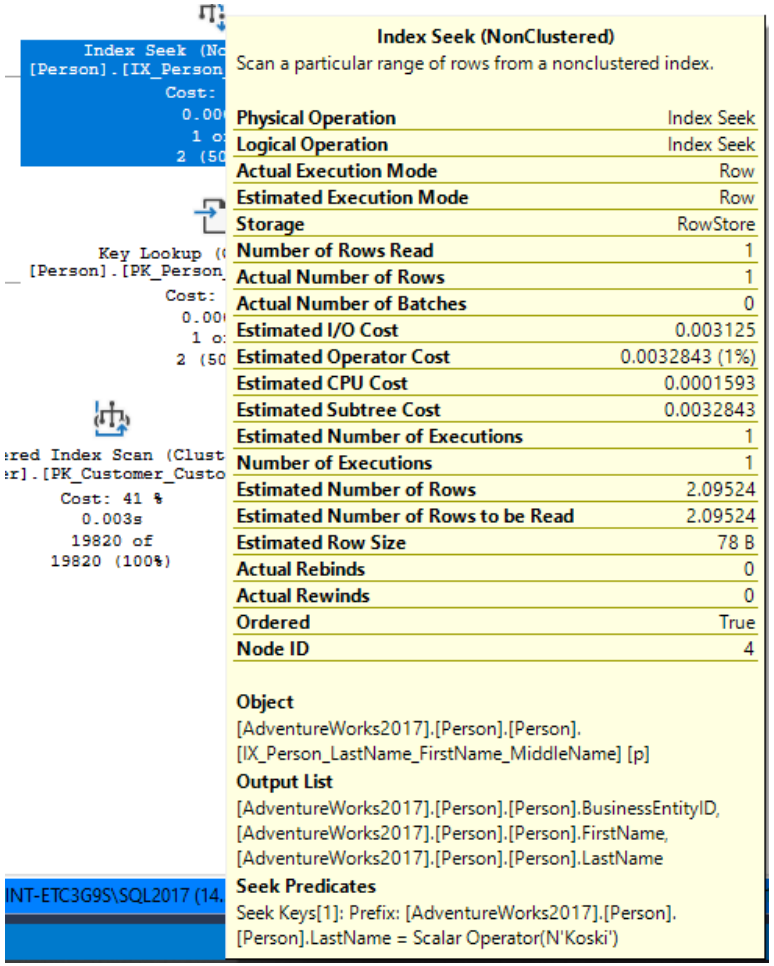
Execution Flow



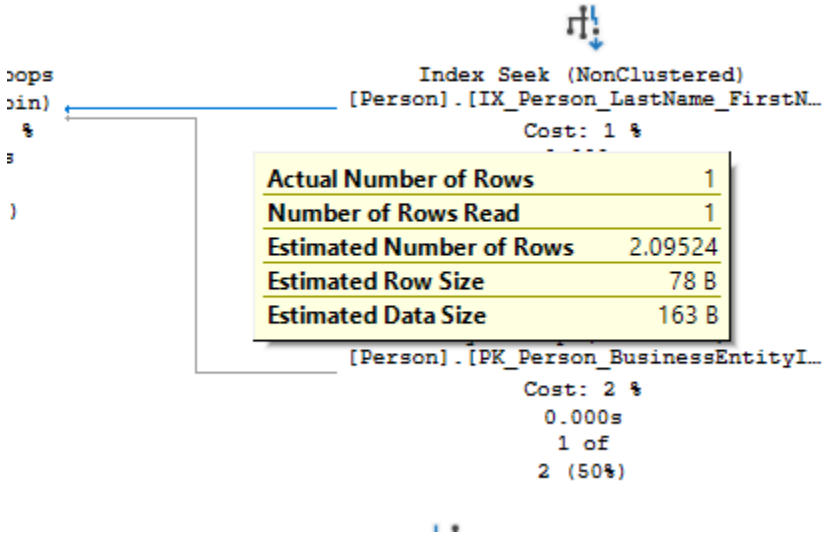
SSMS Graphical Plan

Data flow between the operators and statistical data of each operator

Statistical data for the operator

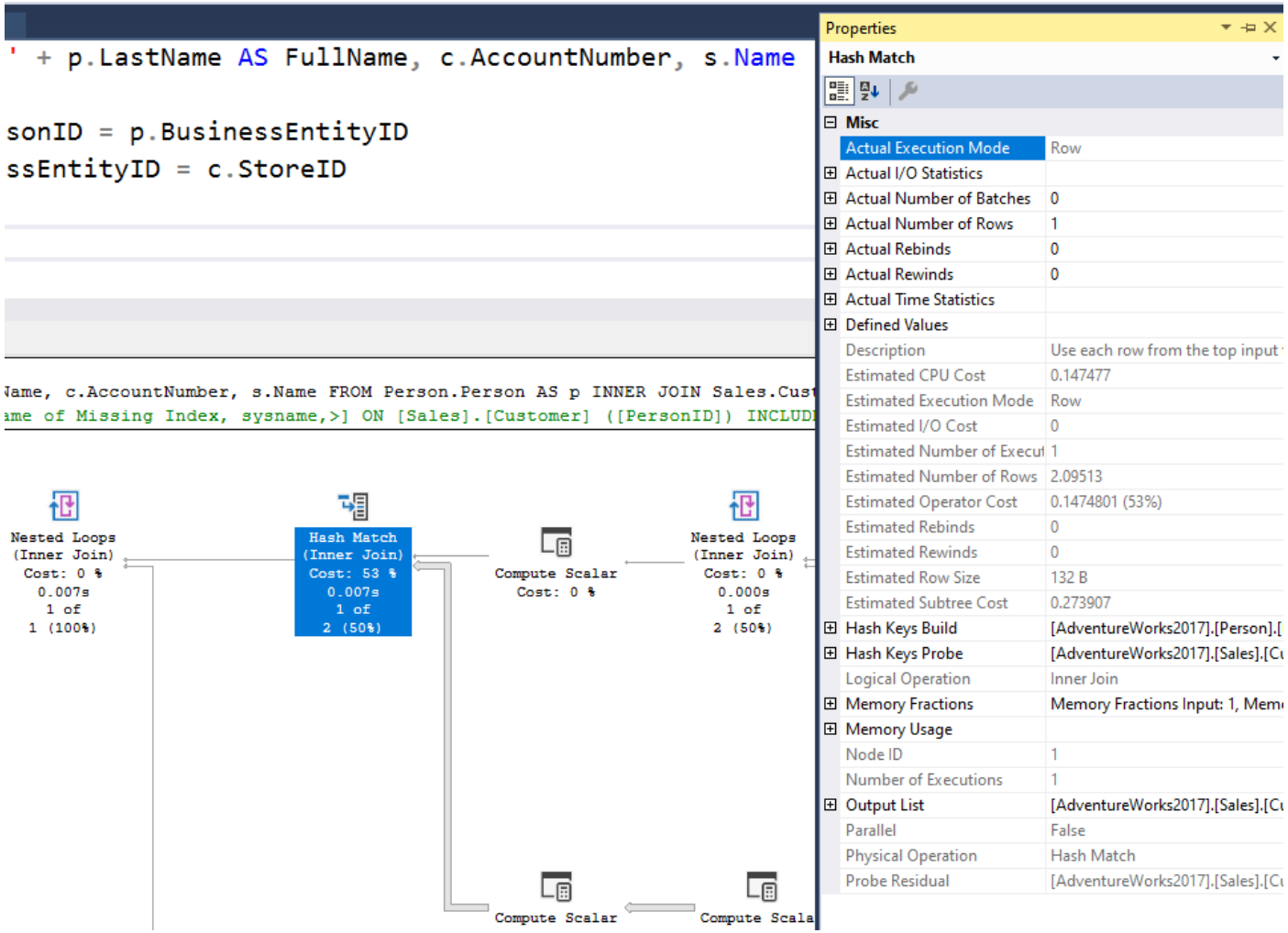


Data flow statistics



SSMS Graphical Plan

Properties sheet



Management Studio Properties sheet includes even more detailed information about each operator and about the overall query plan.

Use the most recent version of Management Studio as every new version display more detailed information about the Query Plan when examining the plan in graphical mode.

SSMS Graphical Plan

Live Query Statistics

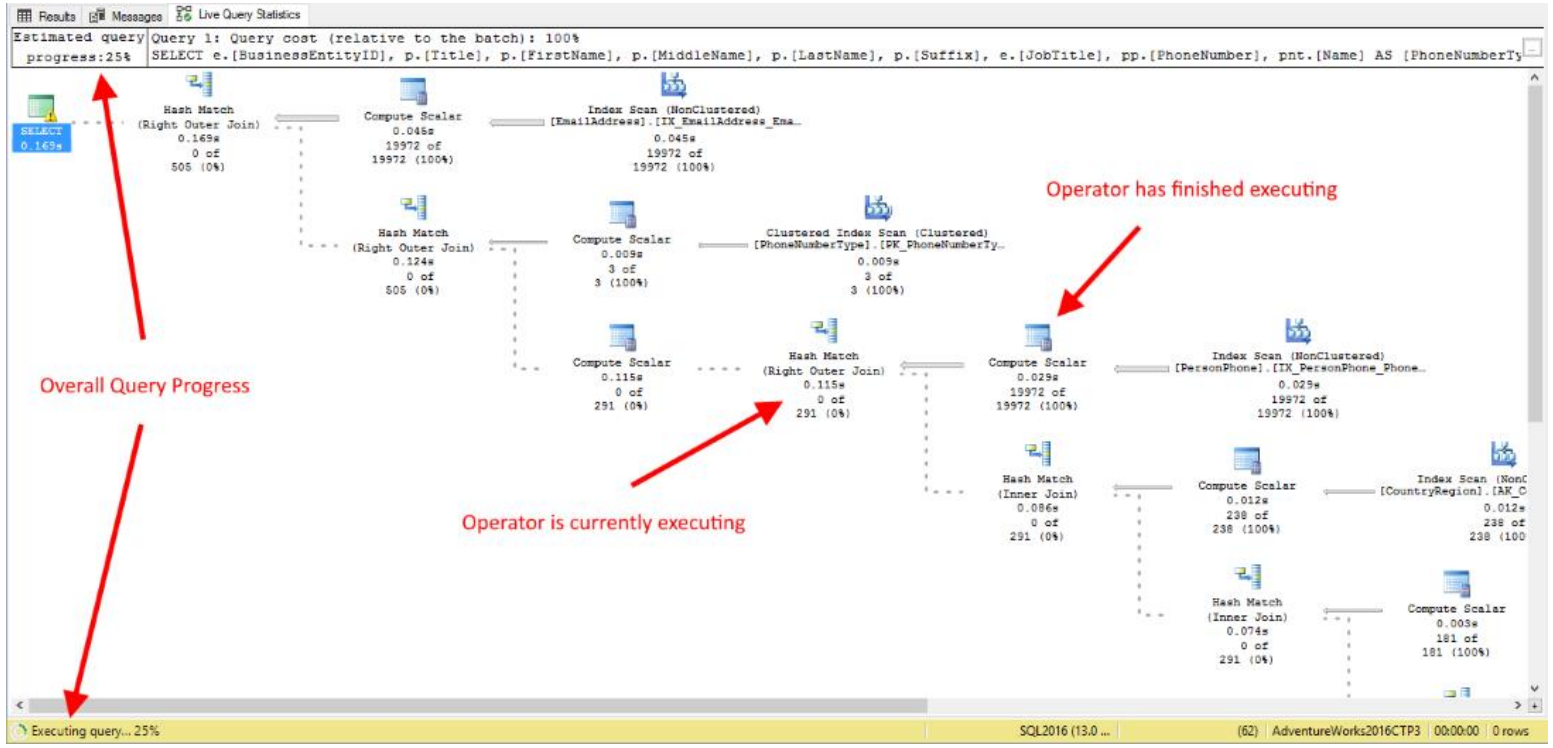
View CPU/memory usage, execution time, and query progress

Allows drill down to live operator-level statistics, such as:

- Number of generated rows
- Elapsed time
- Operator progress
- Live warnings

This feature is primarily intended for troubleshooting purposes.

Using this feature can moderately slow the overall query performance.



Execution Plan

Notable operators

Operators describe how SQL Server executes a query. The query optimizer uses operators to build a query plan to create the result specified in the query.


Table scan


Clustered index scan


Clustered index seek


RID lookup


Key lookup


ColumnStore Index Scan

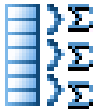

Nonclustered index scan


Nonclustered index seek


Sort


Table spool


Index spool


Stream Aggregation

Execution Plan Join Operators (Code)

```
SELECT SOH.SalesOrderID, SOH.CustomerID,
    OrderQty, UnitPrice, P.Name
FROM SalesLT.SalesOrderHeader as SOH
JOIN SalesLT.SalesOrderDetail AS SOD
ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN SalesLT.Product AS P
ON P.ProductID = SOD.ProductID
```

Execution Plan Join Operators

A Merge Join is useful if both table inputs are in the same sorted order on the same value.

Merge Join
(Inner Join)
Cost: 39 %

A Hash Match is used when the tables being joined are not in the same sorted order.

Hash Match
(Inner Join)
Cost: 47 %

A Nested Loop is use when a small (outer) table is used to lookup a value in a larger (inner) table.

Nested Loops
(Inner Join)
Cost: 3 %

Query Hint Types

Query Hints

- Applied to the whole query using the OPTION clause
- Examples: MAXDOP, RECOMPILE, OPTIMIZE FOR

Table Hints

- Applied to single tables in the query
- Examples: FORCESEEK, FORCESCAN, and INDEX

Join Hints

- Applied to specific joins only
- Examples: LOOP, MERGE, and HASH

Use Hints

- Provides one or more additional hints to the query processor.
- Example: ENABLE_QUERY_OPTIMIZER_HOTFIXES

What to look for in the query plan

Warnings	• Information about possible issues with the plan
Left Most Operator	• Overall properties of the plan to establish baseline
Right to Left	• Solve issues early in the plan
Expensive Operators	• Look from most expensive to least expensive
Sort Operators	• Locate why there is a sort operation and is it needed
Data Flow Statistics	• Thicker arrows mean more data is being passed
Nested Loop Operator	• Possible to create index that covers query
Scans vs Seeks	• Not necessarily bad, but could indicate I/O issues
Skewed Estimates	• Statistics could be stale or invalid

Demonstration

Query Plan Analysis

- Use the graphical execution plan and IO statistics to tune a query.
- Explore Live Query Statistics.

Query Plan Analysis

- Identifying and tuning high cost operators in the query plan

Questions?

Knowledge Check

What are the physical join operators?

What is a method to eliminate a lookup?

Is it recommended to eliminate all lookups operators?

Under what circumstances would a table scan be more efficient than an index seek on a non-clustered, non-covering index?

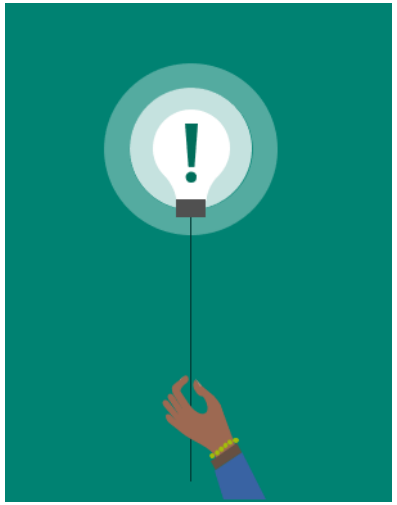
Is a Clustered Index Scan more efficient than a Table Scan?

Lesson 3: SQL Server Query Store

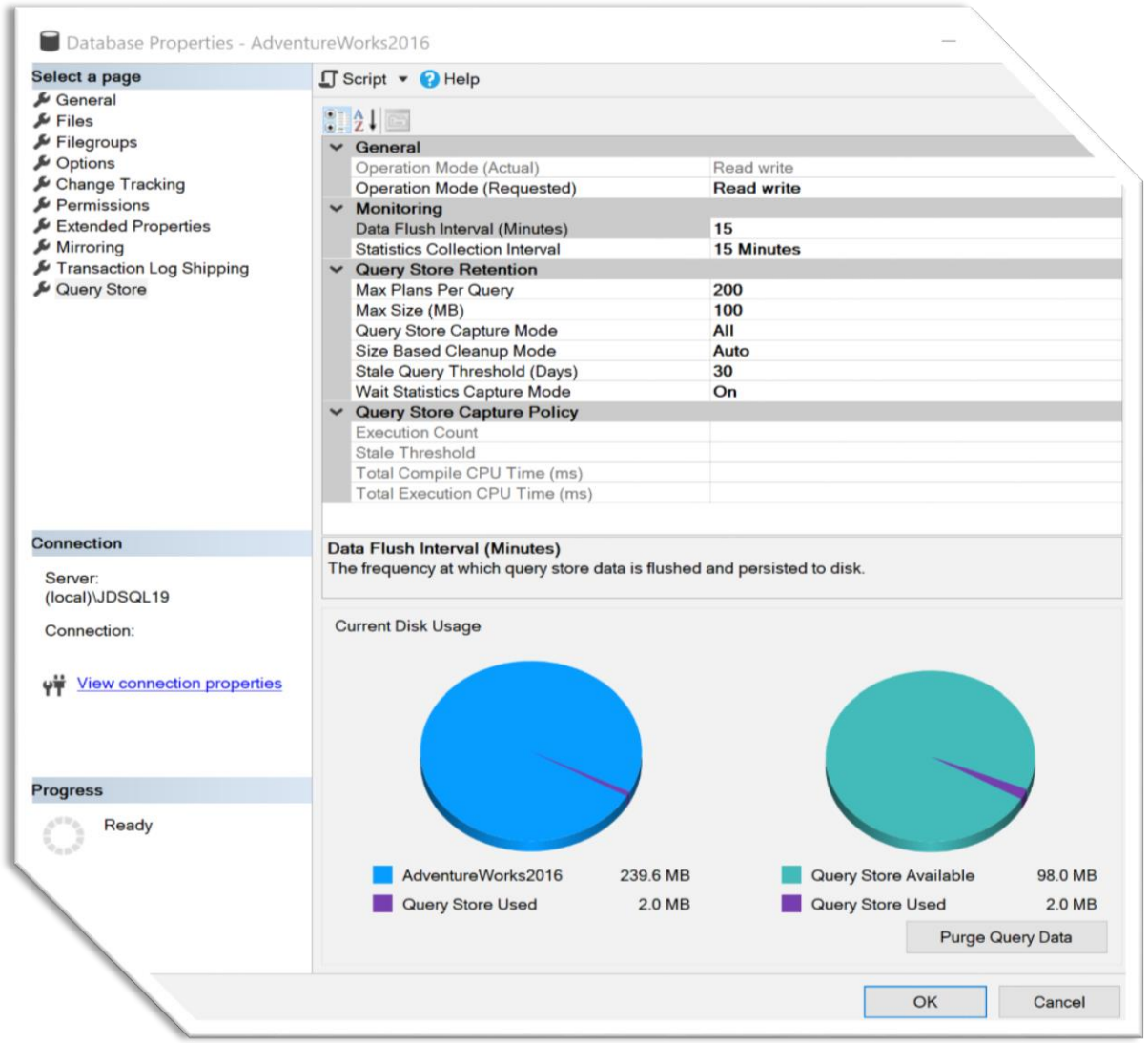
Objectives

After completing this learning, you will be able to:

- Understand what makes Query Store a valuable tool.
- Understand its key usage scenarios.
- Enable Query Store and configure it appropriately.
- List types of runtime data collected by the Query Store.
- Have a basic understanding of the built-in reports.
- Understanding Query Store Hints.



Introducing the Query Store



Query Store is set at the database level

Cannot be used for Master or TempDB system databases but can be enabled for the Model and MSDB system databases.

The user database stores the data in internal tables that can be accessed by using built-in Query Store views.

SQL Server retains this data until the space allocated to Query Store is full or manually purged.

Why use Query Store?

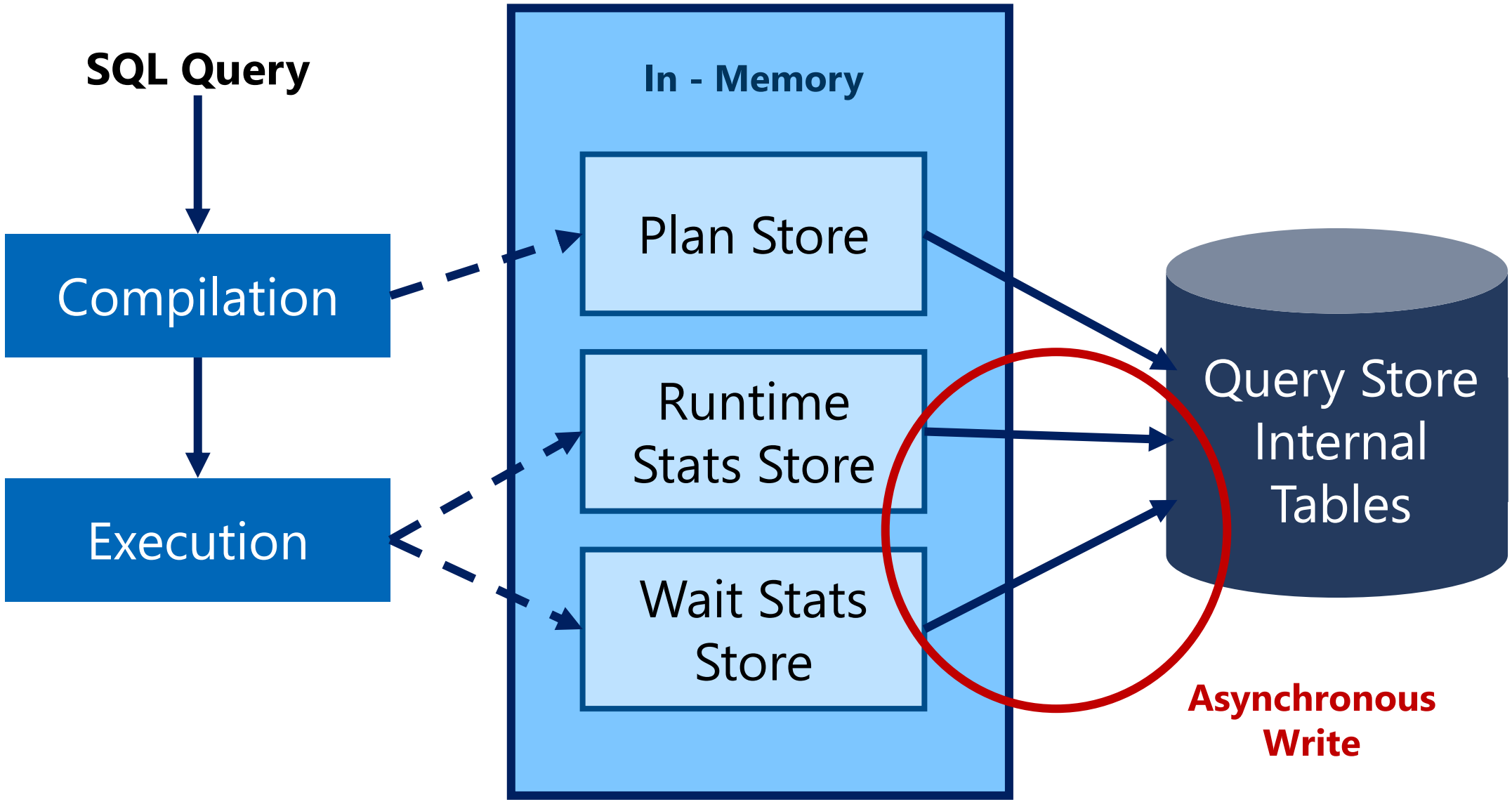
Before Query Store

- Requires manual proactive monitoring to identify execution plan problems.
- Only the latest plan was stored in the procedure cache
- Restart caused data to be lost
- Frequent recompiles of procedures or use of DBCC FREEPROCACHE
- No history or aggregated gathering of data available.

With Query Store

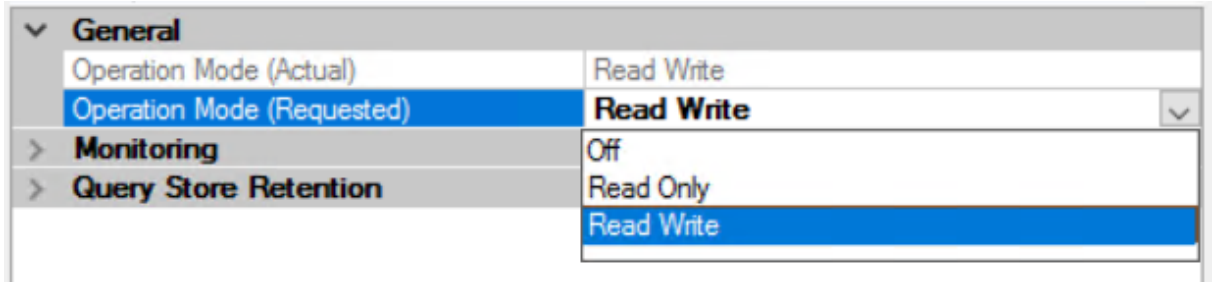
- It stores the history of the execution plans for each query
- It establishes a performance baseline for each plan over time
- It identifies queries that may have regressed
- It is possible to force plans quickly and easily
- It works across server restarts, upgrades, and query recompilation

How Query Store collects and stores data



Query Store Operation Modes

Operation Mode can be set under database properties



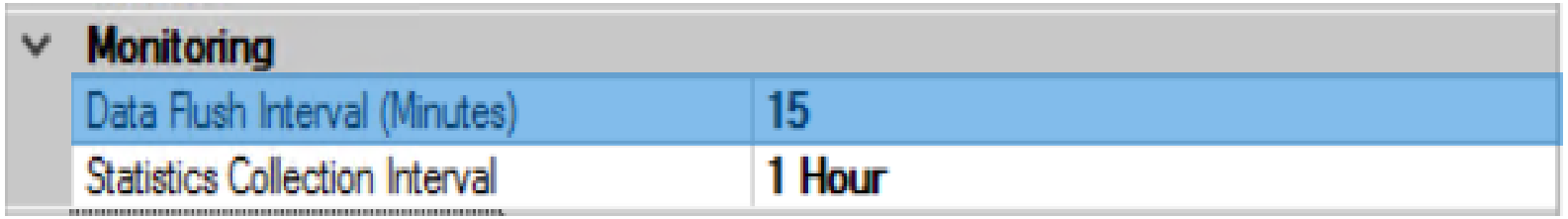
Operation Mode can be enabled two ways using T-SQL. If only using the ON option, the Mode defaults to **Read_Write**

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE = ON;

ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(OPERATION_MODE = READ_WRITE);
```

Query Store Monitoring Settings

Data Flush Interval determines the frequency at which data written to the query store is persisted to disk. (Default is **15 Minutes**).




```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(INTERVAL_LENGTH_MINUTES = 60,
 DATA_FLUSH_INTERVAL_SECONDS = 900 );
```

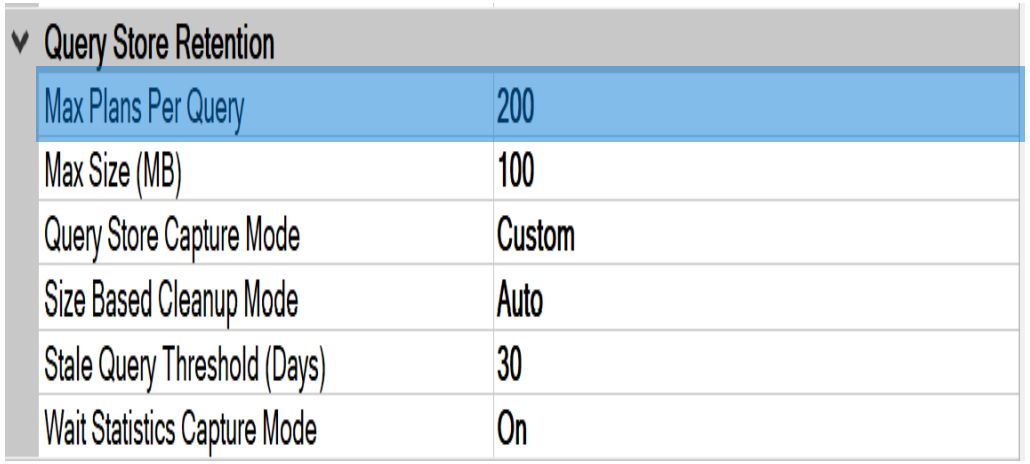
Query Store Monitoring Settings

Statistics Collection Interval determines the time interval at which runtime execution statistics data is aggregated into the query store. Only the values of 1, 5, 10, 15, 60, and 1440 minutes is allowed. (Default is **60**).


```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(INTERVAL_LENGTH_MINUTES = 60,
 DATA_FLUSH_INTERVAL_SECONDS = 900 );
```

Query Store Retention Settings

Max Plans Per Query is a new retention setting introduced in SQL Server 2017 and is an integer representing the maximum number of plans maintained for each query. (Default is **200**).

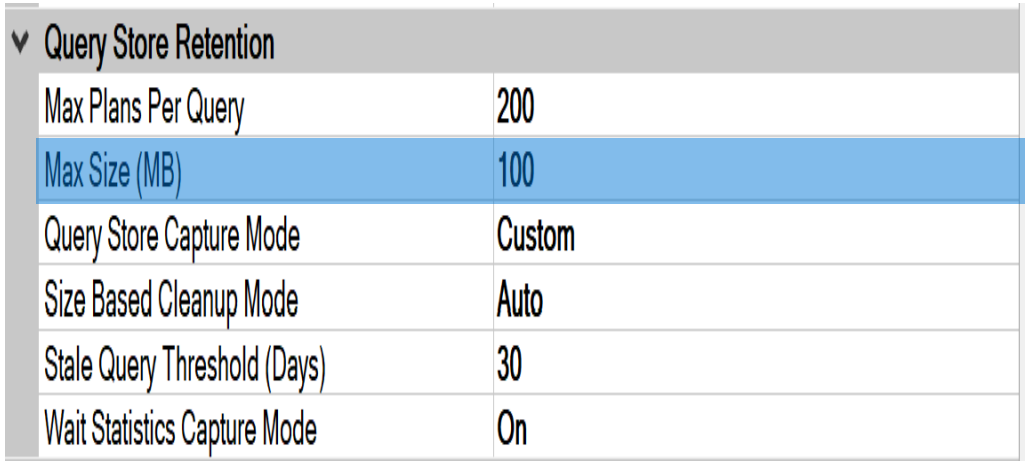


Query Store Retention	
Max Plans Per Query	200
Max Size (MB)	100
Query Store Capture Mode	Custom
Size Based Cleanup Mode	Auto
Stale Query Threshold (Days)	30
Wait Statistics Capture Mode	On

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(MAX_PLANS_PER_QUERY = 20,
 MAX_STORAGE_SIZE_MB = 1000,
 QUERY_CAPTURE_MODE = CUSTOM,
 SIZE_BASED_CLEANUP_MODE = AUTO,
 CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 90,
 WAIT_STATS_CAPTURE_MODE = ON);
 GO
```

Query Store Retention Settings

Max Size (MB) configures the maximum storage size for the query store. (Default is **100MB**) When the query store limit is reached, query store changes the state from read-write to read-only.

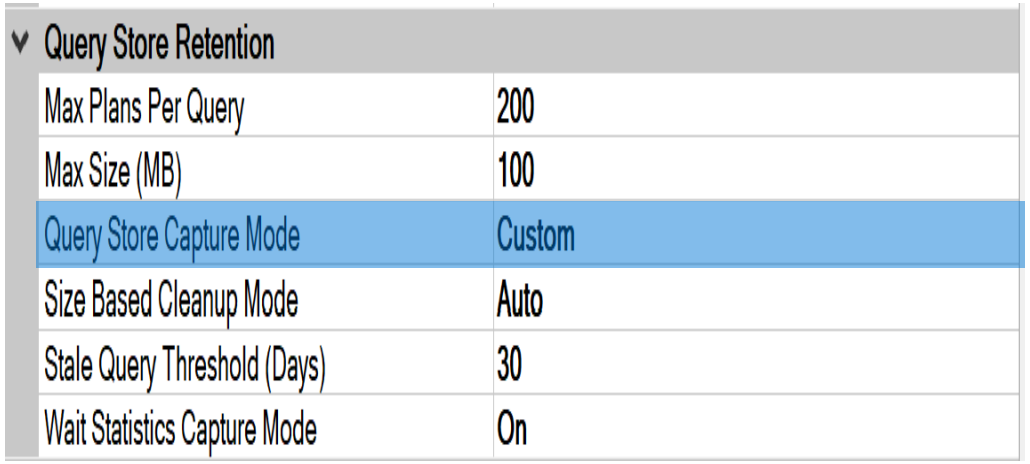


Query Store Retention	
Max Plans Per Query	200
Max Size (MB)	100
Query Store Capture Mode	Custom
Size Based Cleanup Mode	Auto
Stale Query Threshold (Days)	30
Wait Statistics Capture Mode	On

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(MAX_PLANS_PER_QUERY = 20,
 MAX_STORAGE_SIZE_MB = 1000,
 QUERY_CAPTURE_MODE = CUSTOM,
 SIZE_BASED_CLEANUP_MODE = AUTO,
 CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 90,
 WAIT_STATS_CAPTURE_MODE = ON);
 GO
```

Query Store Retention Settings

Query Store Capture Mode determines to capture all the queries (Default is **ALL**), or relevant queries based on execution count and resource consumption (**AUTO**) or stop capturing queries (**NONE**). SQL Server 2019 introduces an additional (**CUSTOM**) setting.

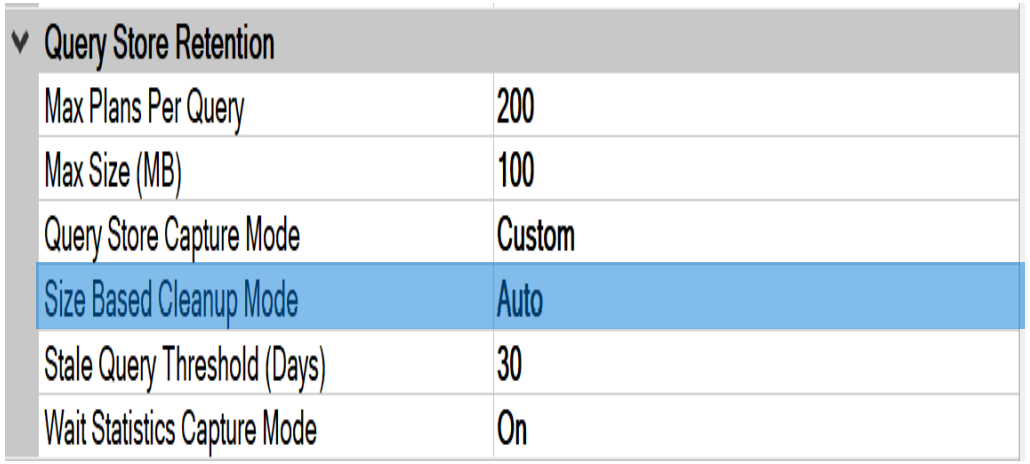


Query Store Retention	
Max Plans Per Query	200
Max Size (MB)	100
Query Store Capture Mode	Custom
Size Based Cleanup Mode	Auto
Stale Query Threshold (Days)	30
Wait Statistics Capture Mode	On

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(MAX_PLANS_PER_QUERY = 20,
 MAX_STORAGE_SIZE_MB = 1000,
 QUERY_CAPTURE_MODE = CUSTOM,
 SIZE_BASED_CLEANUP_MODE = AUTO,
 CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 90,
 WAIT_STATS_CAPTURE_MODE = ON);
 GO
```

Query Store Retention Settings

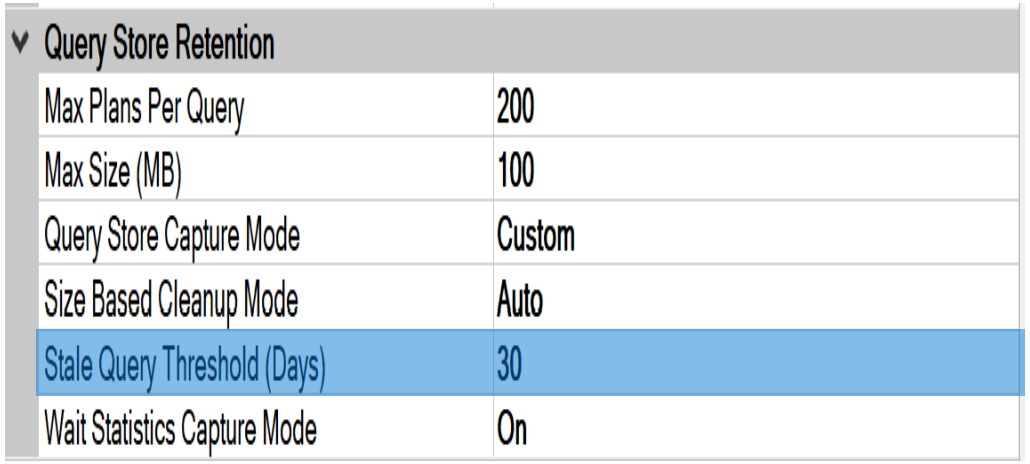
Size Based Cleanup Mode determines whether the cleanup process will be automatically activated when the total amount of data gets close to the maximum size. (Default is **Auto**).




```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(MAX_PLANS_PER_QUERY = 20,
 MAX_STORAGE_SIZE_MB = 1000,
 QUERY_CAPTURE_MODE = CUSTOM,
 SIZE_BASED_CLEANUP_MODE = AUTO,
 CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 90,
 WAIT_STATS_CAPTURE_MODE = ON);
 GO
```

Query Store Retention Settings

Stale Query Threshold (Days) determines the number of days to retain data in the query store. (Default is **30 days** and Maximum is **367 days**).



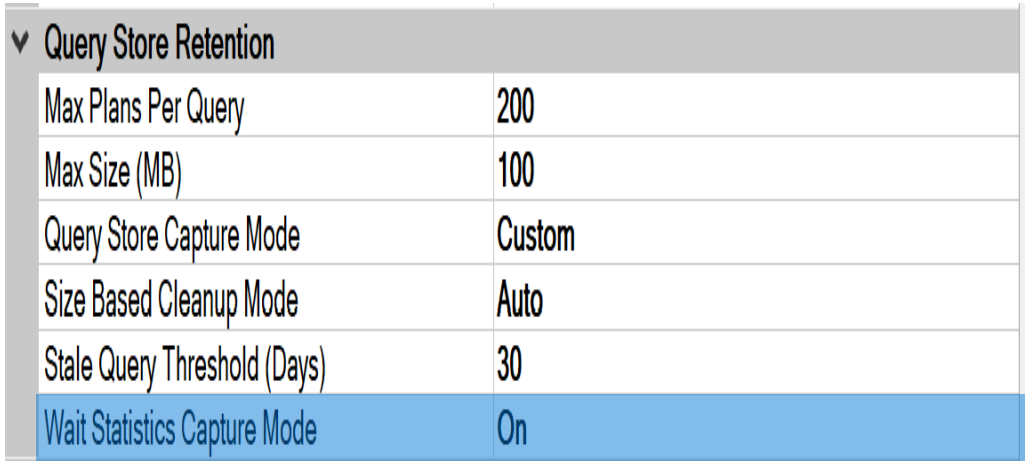
Query Store Retention	
Max Plans Per Query	200
Max Size (MB)	100
Query Store Capture Mode	Custom
Size Based Cleanup Mode	Auto
Stale Query Threshold (Days)	30
Wait Statistics Capture Mode	On

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(MAX_PLANS_PER_QUERY = 20,
 MAX_STORAGE_SIZE_MB = 1000,
 QUERY_CAPTURE_MODE = CUSTOM,
 SIZE_BASED_CLEANUP_MODE = AUTO,
 CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 90,
 WAIT_STATS_CAPTURE_MODE = ON);
 GO
```

Query Store Retention Settings

Wait Statistics Capture Mode is a new retention setting introduced in SQL Server 2017 that controls if Query Store captures wait statistics information.

(Default = **ON**).



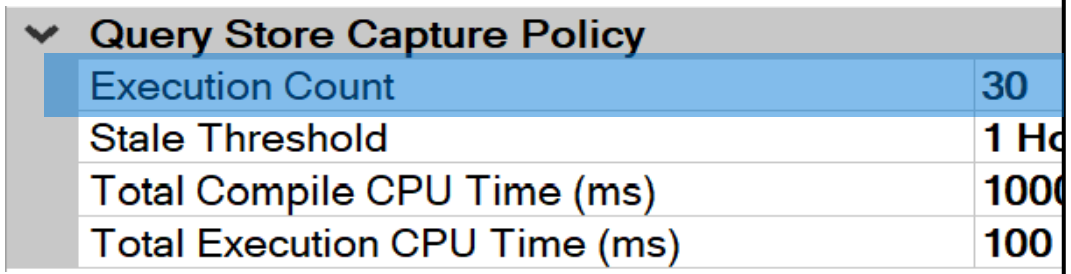
Query Store Retention	
Max Plans Per Query	200
Max Size (MB)	100
Query Store Capture Mode	Custom
Size Based Cleanup Mode	Auto
Stale Query Threshold (Days)	30
Wait Statistics Capture Mode	On

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(MAX_PLANS_PER_QUERY = 20,
 MAX_STORAGE_SIZE_MB = 1000,
 QUERY_CAPTURE_MODE = CUSTOM,
 SIZE_BASED_CLEANUP_MODE = AUTO,
 CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 90,
 WAIT_STATS_CAPTURE_MODE = ON);
 GO
```

Query Store Capture Policy Settings

Introduced in SQL Server 2019 and available if the Query Store Capture Mode setting has been set to **CUSTOM.**

The value for the **EXECUTION COUNT** is the value a query must exceed within the Stale Threshold time period to be captured by the Query Store.




```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(QUERY_CAPTURE_POLICY =
(EXECUTION_COUNT = 100,
 STALE_CAPTURE_POLICY_THRESHOLD = 24 HOURS,
 TOTAL_COMPILE_CPU_TIME_MS = 10000,
 TOTAL_EXECUTION_CPU_TIME_MS = 20000);
 GO
```

Query Store Capture Policy Settings

Introduced in SQL Server 2019 and available if the Query Store Capture Mode setting has been set to **CUSTOM.**

The value for the **Stale Threshold** can be from 1 hour up to 7 days. This setting specifies the time given to exceed the values of the three other settings for a query to be captured.

Query Store Capture Policy	
Execution Count	30
Stale Threshold	1 Hour
Total Compile CPU Time (ms)	1000
Total Execution CPU Time (ms)	100

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(QUERY_CAPTURE_POLICY =
(EXECUTION_COUNT = 100,
 STALE_CAPTURE_POLICY_THRESHOLD = 24 HOURS,
 TOTAL_COMPILE_CPU_TIME_MS = 10000,
 TOTAL_EXECUTION_CPU_TIME_MS = 20000);
 GO
```

Query Store Capture Policy Settings

Introduced in SQL Server 2019 and available if the Query Store Capture Mode setting has been set to **CUSTOM.**

The value for the **Total Compile CPU Time (ms)** is the value in milliseconds that a query must exceed within the **Stale Threshold** time period to be captured by the Query Store.

Query Store Capture Policy	
Execution Count	30
Stale Threshold	1 Hour
Total Compile CPU Time (ms)	1000
Total Execution CPU Time (ms)	100

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(QUERY_CAPTURE_POLICY =
(EXECUTION_COUNT = 100,
 STALE_CAPTURE_POLICY_THRESHOLD = 24 HOURS,
 TOTAL_COMPILE_CPU_TIME_MS = 10000,
 TOTAL_EXECUTION_CPU_TIME_MS = 20000);
 GO
```

Query Store Capture Policy Settings

Introduced in SQL Server 2019 and available if the Query Store Capture Mode setting has been set to **CUSTOM.**

The value for the **Total Execution CPU Time (ms)** is the value in milliseconds that a query must exceed within the **Stale Threshold** time period to be captured by the Query Store.

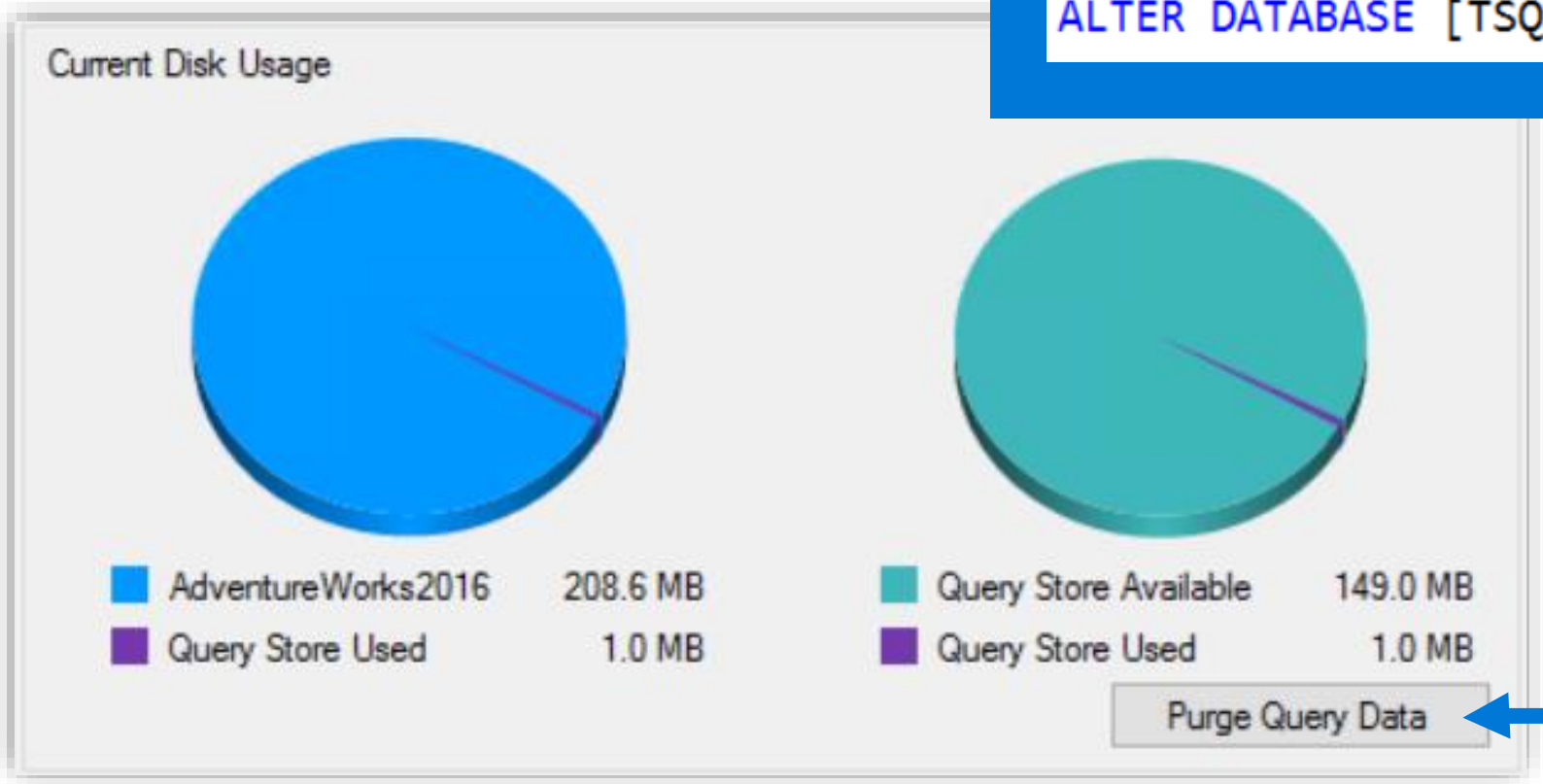
Query Store Capture Policy	
Execution Count	30
Stale Threshold	1 Hour
Total Compile CPU Time (ms)	1000
Total Execution CPU Time (ms)	100

```
ALTER DATABASE [AdventureWorksPTO] SET QUERY_STORE
(QUERY_CAPTURE_POLICY =
(EXECUTION_COUNT = 100,
 STALE_CAPTURE_POLICY_THRESHOLD = 24 HOURS,
 TOTAL_COMPILE_CPU_TIME_MS = 10000,
 TOTAL_EXECUTION_CPU_TIME_MS = 20000);
 GO
```

Purge Query Data

Data can be manually purged from the Query Store.

```
ALTER DATABASE [TSQL] SET QUERY_STORE CLEAR;
```

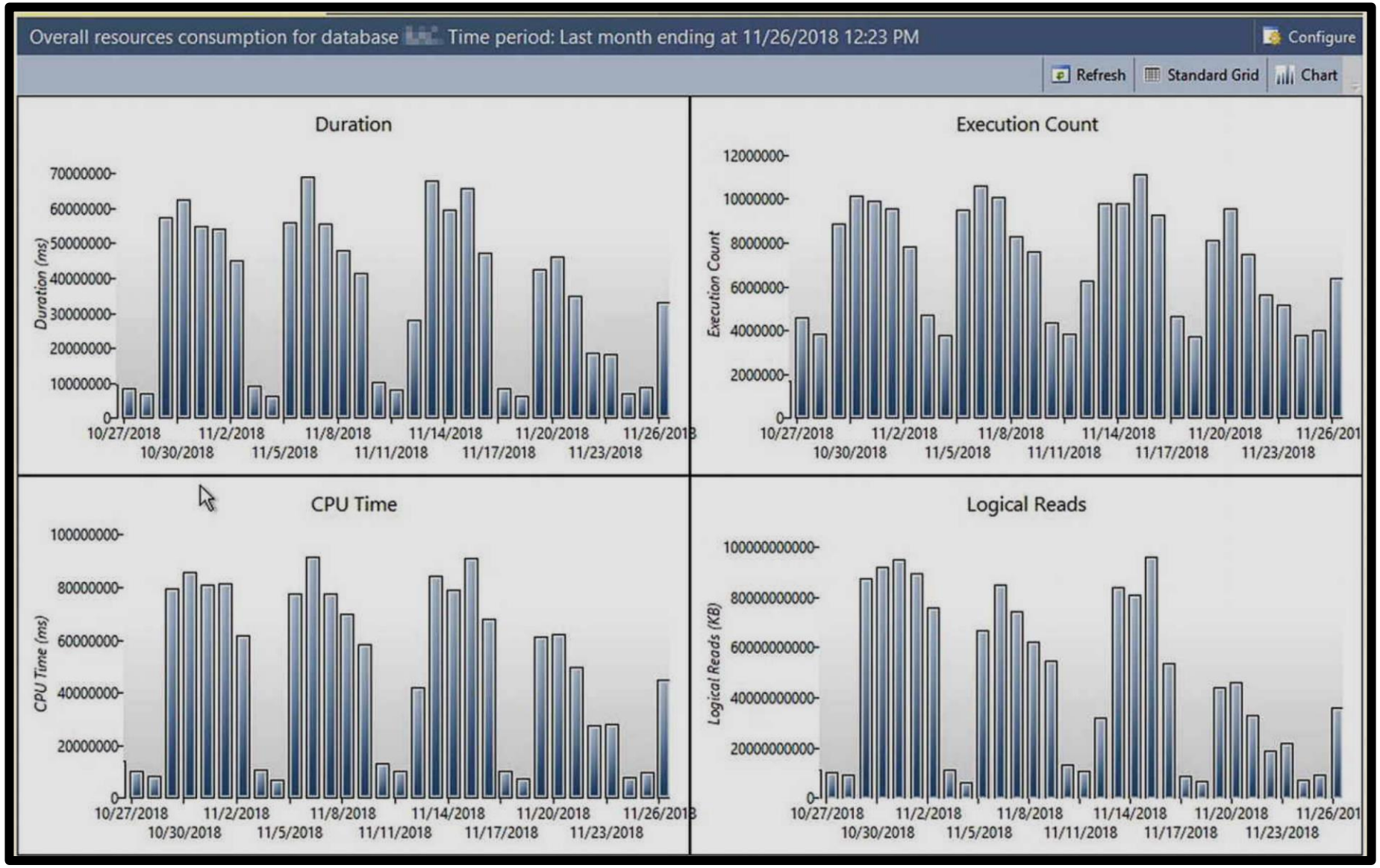


Built-in Reports

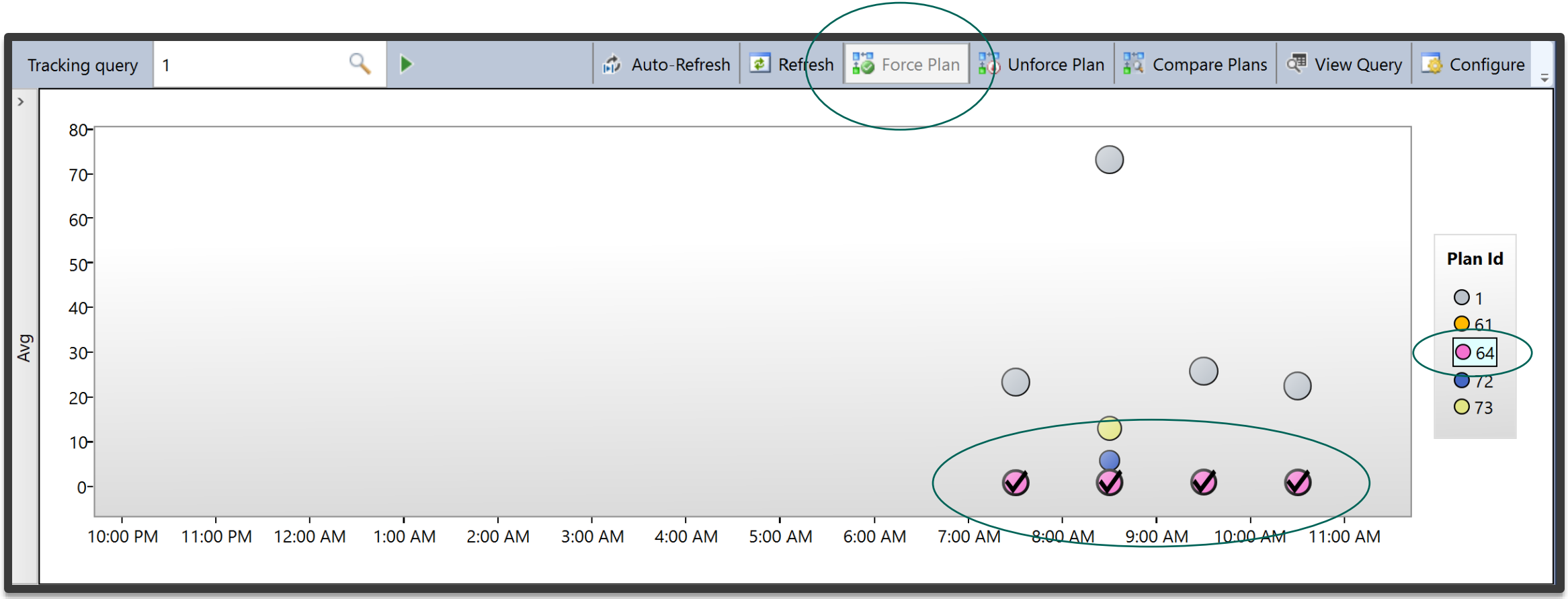
Regressed queries	• Automatic detection of queries that have begun executing more slowly
Overall resource consumption	• Historic view across 4 performance metrics of your choice
Top Resource-consuming queries	• Which queries are the costliest to execute
Queries with forced plans	• Check on the performance of queries using forced plans
Queries with high variation	• Inconsistently performing queries may need tuning
Query wait statistics	• Identify performance bottlenecks
Tracked queries	• Focus on plans and metrics for a single query

```
-- Permission to view reports
GRANT VIEW DATABASE STATE TO <UserName>;
```

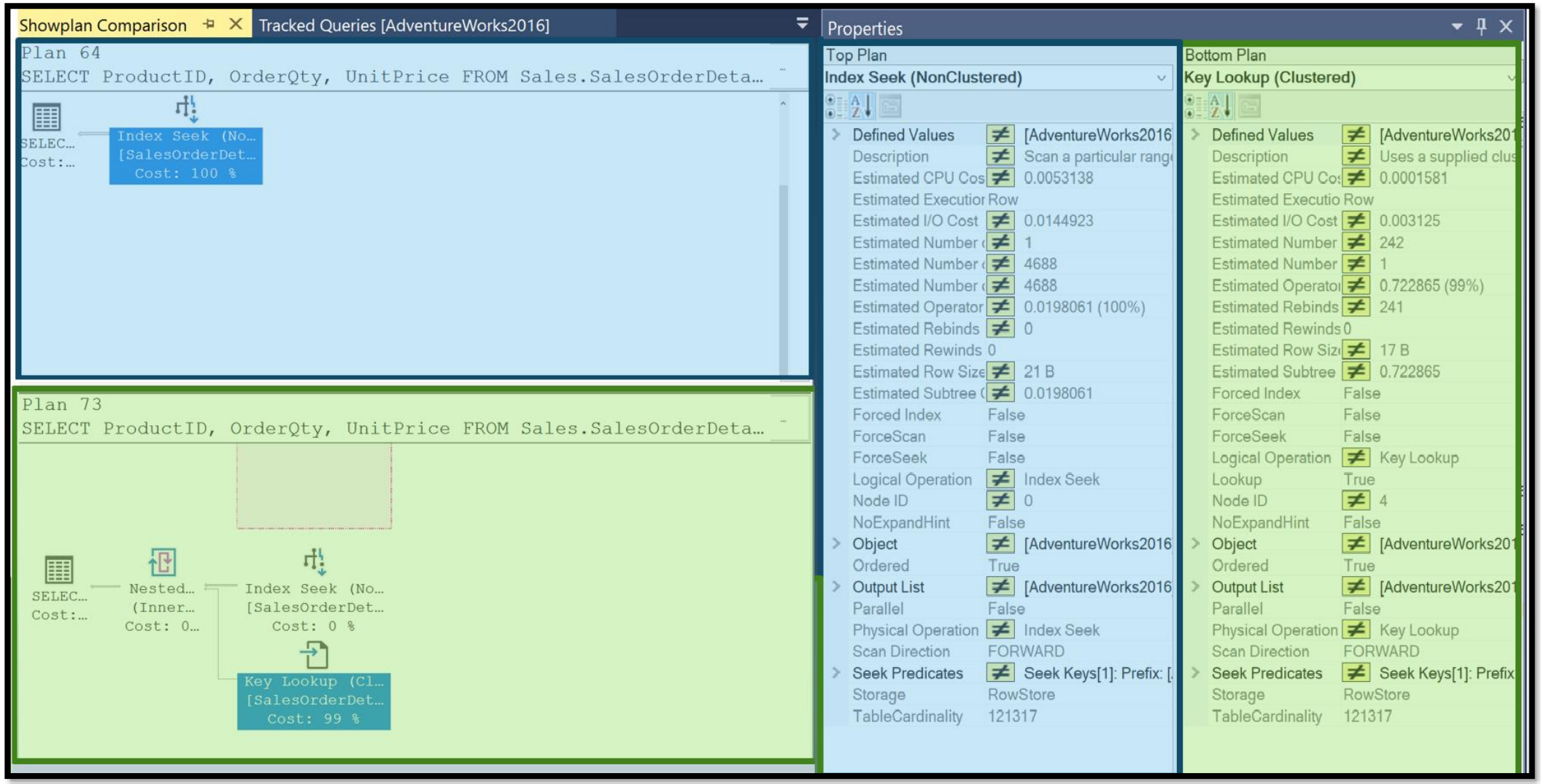
Establishing a Baseline



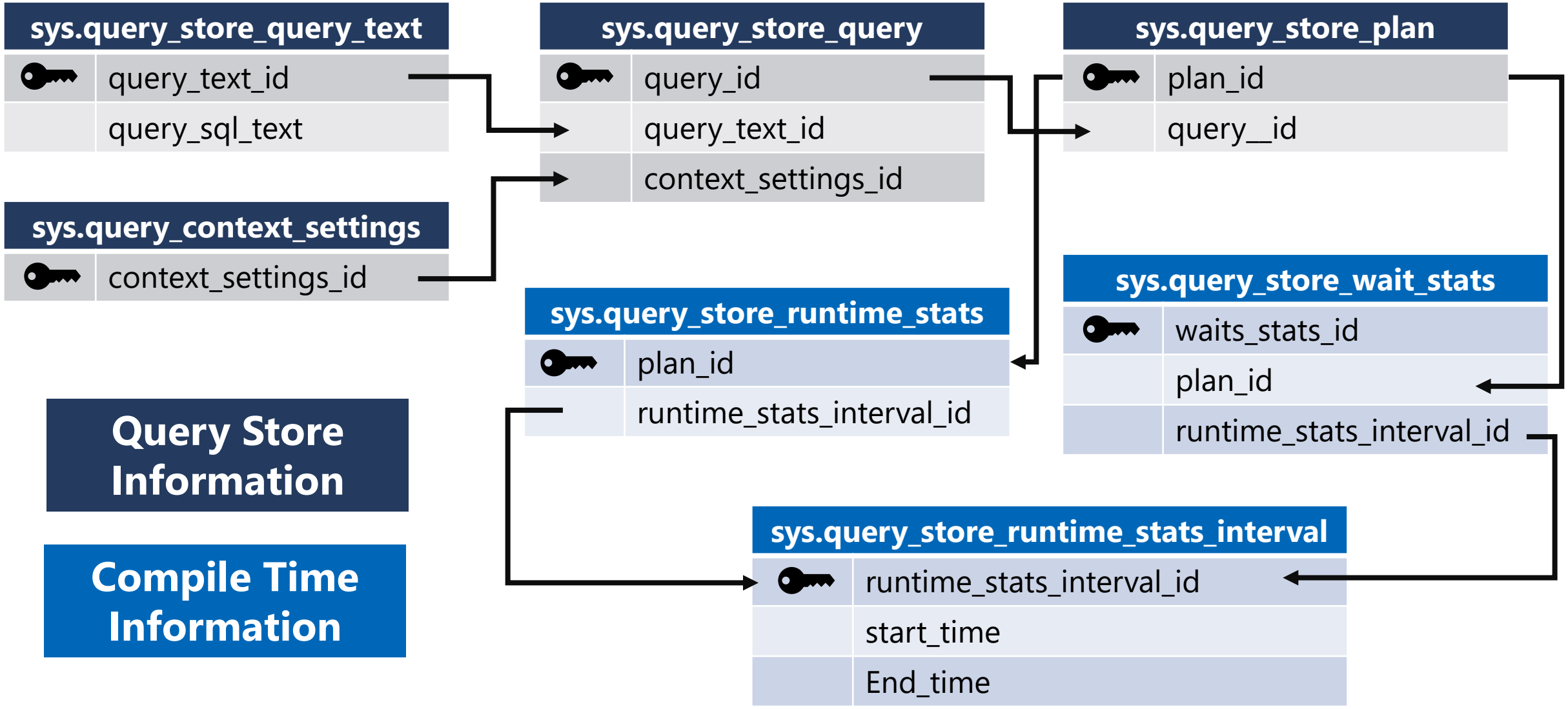
Force Plan


Tracking query
1
Auto-Refresh
Refresh
Force Plan
Unforce Plan
Compare Plans
View Query
Configure
Avg
80
70
60
50
40
30
20
10
0
10:00 PM
11:00 PM
12:00 AM
1:00 AM
2:00 AM
3:00 AM
4:00 AM
5:00 AM
6:00 AM
7:00 AM
8:00 AM
9:00 AM
10:00 AM
11:00 AM
Plan Id
1
61
64
72
73

Plan Compare

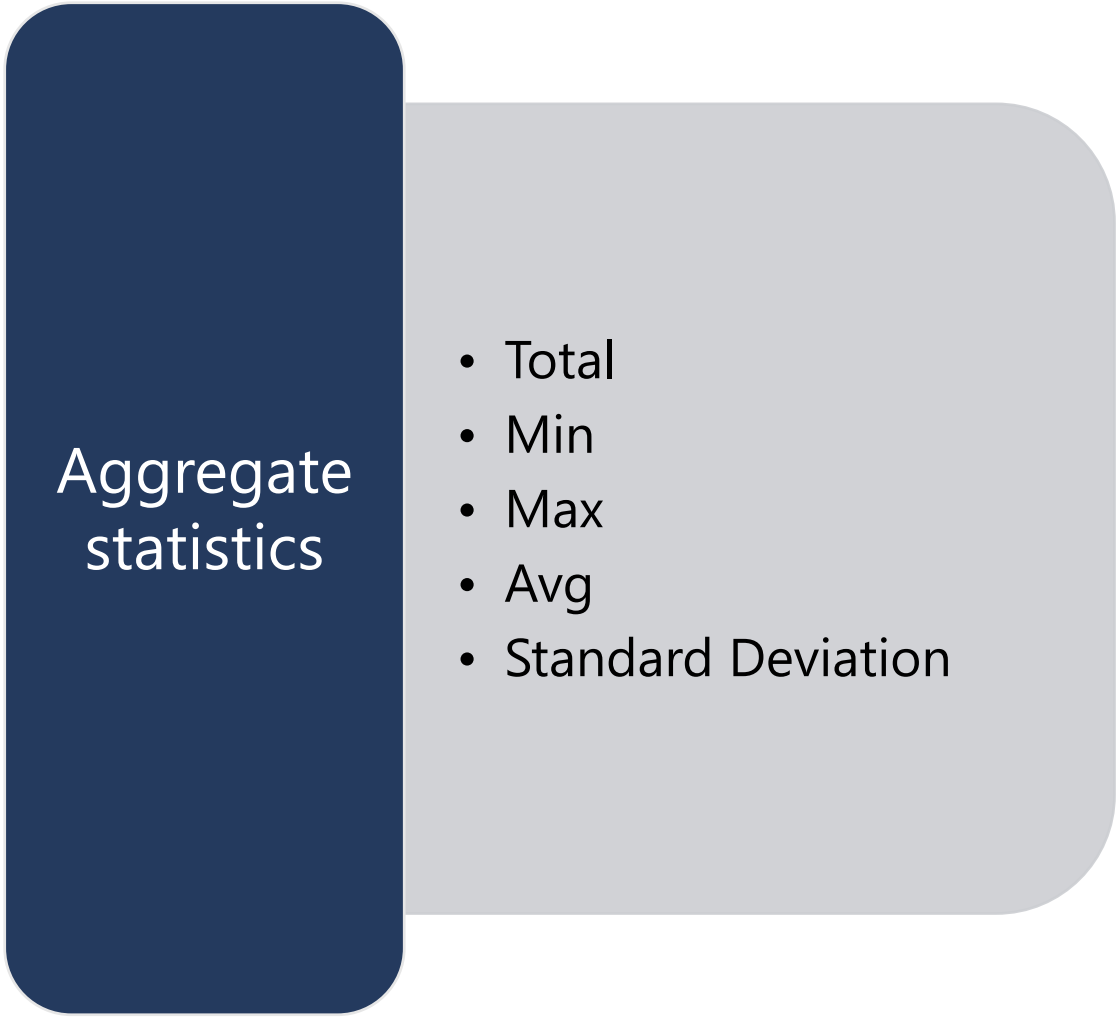


Query Store Catalog Views



Runtime Metrics and Statistics

- Execution count
- Duration
- CPU
- Logical reads
- Logical writes
- Physical reads
- CLR Time
- DOP
- Memory consumption
- Row Count
- Log memory used
- Tempdb memory used
- Wait time



Using Query Store Catalog Views

Finding the TOP 10 most frequently executed SQL Server Queries in the Query Store.

```
SELECT TOP 10 t.query_sql_text, q.query_id
FROM sys.query_store_query_text as t
JOIN sys.query_store_query as q
ON t.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
WHERE rs.count_executions >1
GROUP BY t.query_sql_text, q.query_id
ORDER BY SUM(rs.count_executions)
```

Query Store read replica support for Availability Groups

New feature in SQL Server 2022 – currently in preview

Execution metrics for queries run on secondary replicas

Data is sent from secondaries back to the primary replica

Persisted in the primary replica's Query Store

You must enable trace flag 12606 before you can enable Query Store for secondary replicas.

Considerations

- Sharing bandwidth with outgoing transaction records
- A shared Query Store will be larger
- Impact of *ad hoc* workloads run on secondary replicas

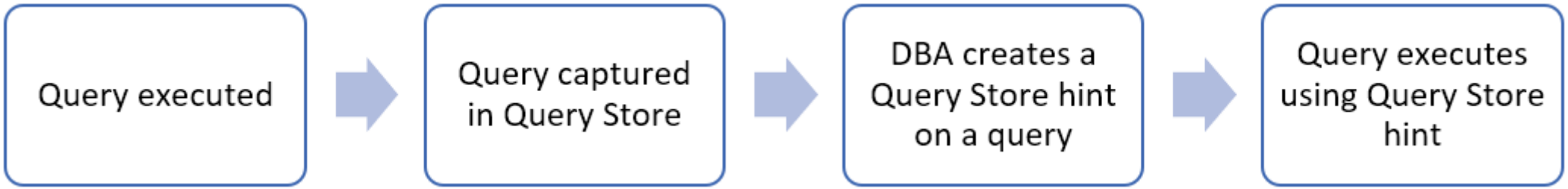
Query Store hints

Query Store hints are available in Azure SQL Database and Azure SQL Managed Instance. Query Store hints are also a feature introduced to SQL Server in SQL Server 2022 (16.x).

As the name suggests, this feature extends and depends on the **Query Store**.

Not all query hints are supported. Here is a list of **Supported query hints.**

Because the SQL Server Query Optimizer typically selects the best execution plan for a query, we recommend only using hints as a last resort. For more information, **Query Hints**.



Query Store hints – Use Cases

Use Cases

- When code can't be changed
- Override other hints/plan guides
- Recompile a query on each execution.
- Cap the memory grant size for a bulk insert operation.
- Limit the maximum degree of parallelism when updating statistics.
- Use a Hash join instead of a Nested Loops join.
- Use compatibility level 110 for a specific query while keeping everything else in the database at compatibility level 150.

Setting, clearing, and viewing query store hints.

```
--First identify the Query Store query_id of the query statement you with to modify.

-- Adding a query store hints.
EXEC sys.sp_query_store_set_hints @query_id = 51, @query_hints = N'OPTION(RECOMPILE)';

--Updating or adding additional query store hints.
EXEC sys.sp_query_store_set_hints @query_id = 51,
@query_hints = N'OPTION(RECOMPILE, MAXDOP 8, USE HINT(''DISALLOW_BATCH_MODE''))';

--Removing query store hints.
EXEC sys.sp_query_store_clear_hints @query_id = 51;

--Viewing configured query store hints.
SELECT * FROM sys.query_store_query_hints
```

Troubleshooting Using the Query Store

LAB

Questions?

Knowledge Check

If upgrading from SQL Server 2012 to 2019. Which report should figure prominently in your upgrade plans?

In a report's bar chart what does each bar represent?

Which report can help troubleshoot a parameter sniffing issue?

Querying the wait statistics DMV it returns high PAGEIOLATCH waits. Which report can help identify queries with high IO wait times?

Someone has dropped an index needed by a forced plan. What happens the next time the query executes? What happens if the index is recreated?

Questions?