ON TOUR | NEW YORK
PASS

Magnificent Seven and Beyond: Intelligent Query Processing in SQL Server

Taiob Ali
He/Him/His

Database Solutions Manager

GMO LLC

Taiob Ali

He/Him

Database Solutions Manager
GMO LLC

I am a Microsoft Data Platform MVP with over 19 years of experience designing and implementing data solutions across finance, e-commerce, and healthcare. My expertise encompasses the Microsoft Data Platform, MongoDB, Azure AI, and Python, enabling data-driven innovation.

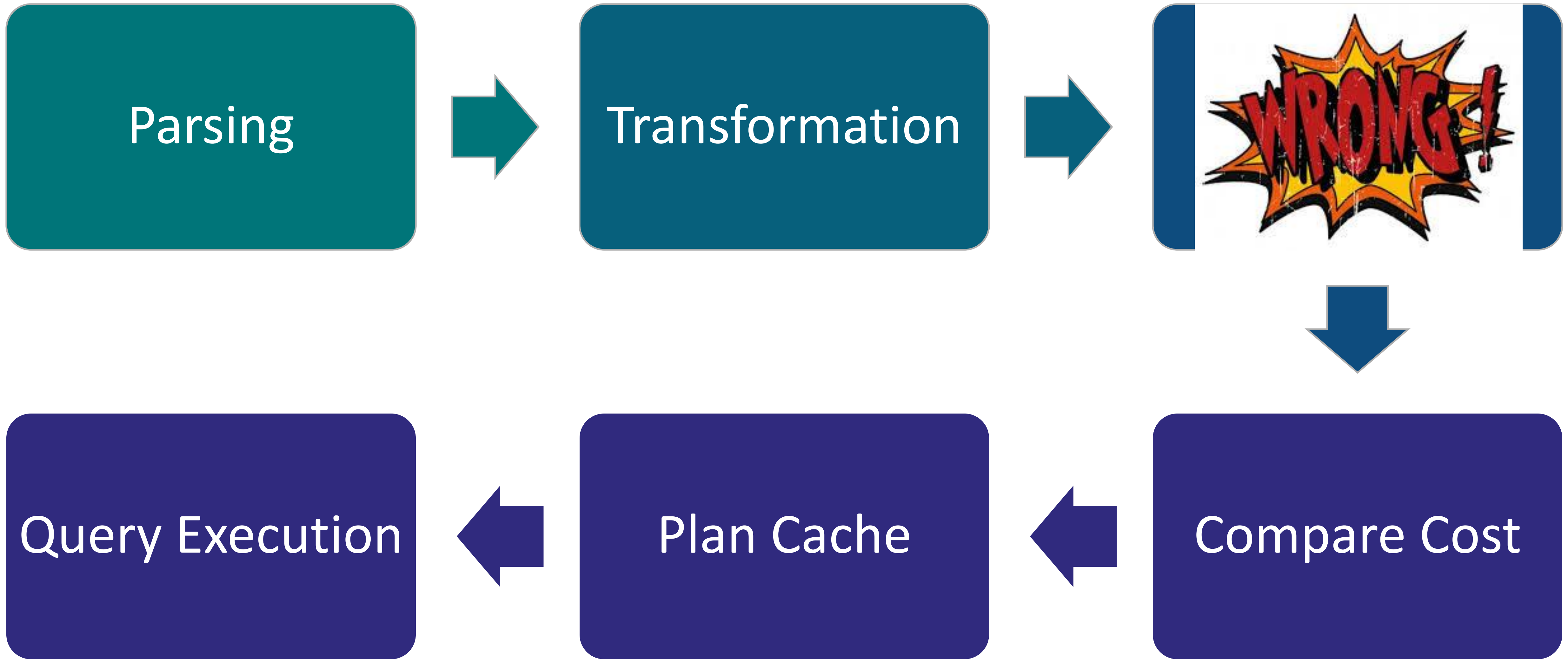
As a dedicated community advocate, I've presented at over 100 events worldwide, including SQL Saturdays, Data Saturdays, and international conferences. I founded the Database Professionals Virtual Meetup Group, serve on the New England SQL Server User Group, and the SQL Saturday boards.

@sqlworldwide
sqlworldwide
http://www.sqlworldwide.com/

Your feedback is important to us

Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation


Parsing
Transformation
WRONG!
Compare Cost
Plan Cache
Query Execution

Missing statistics

Stale statistics

Sample rate (200 steps only)

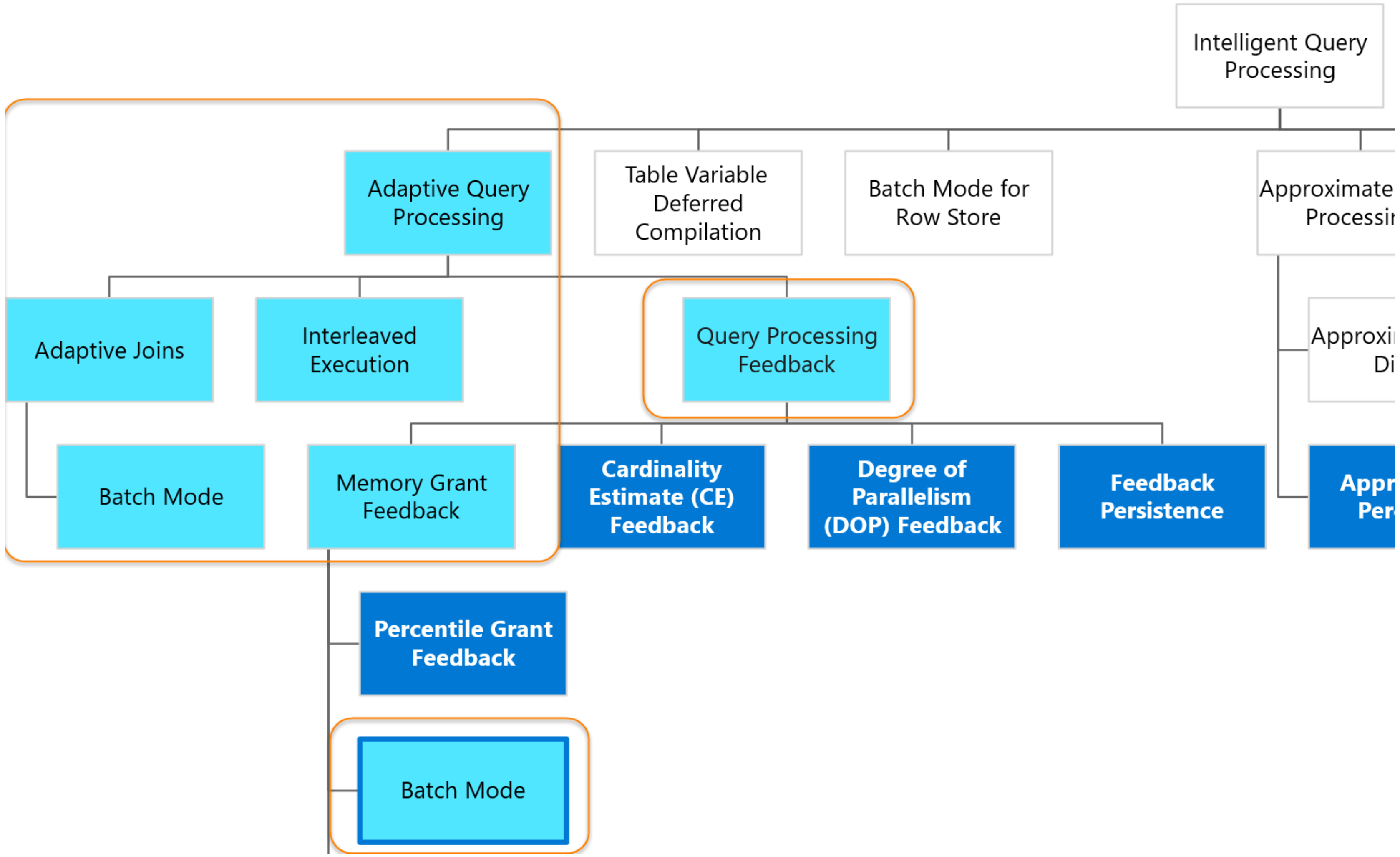
Parameter sniffing

Out-of-model query constructs

Correlation assumption

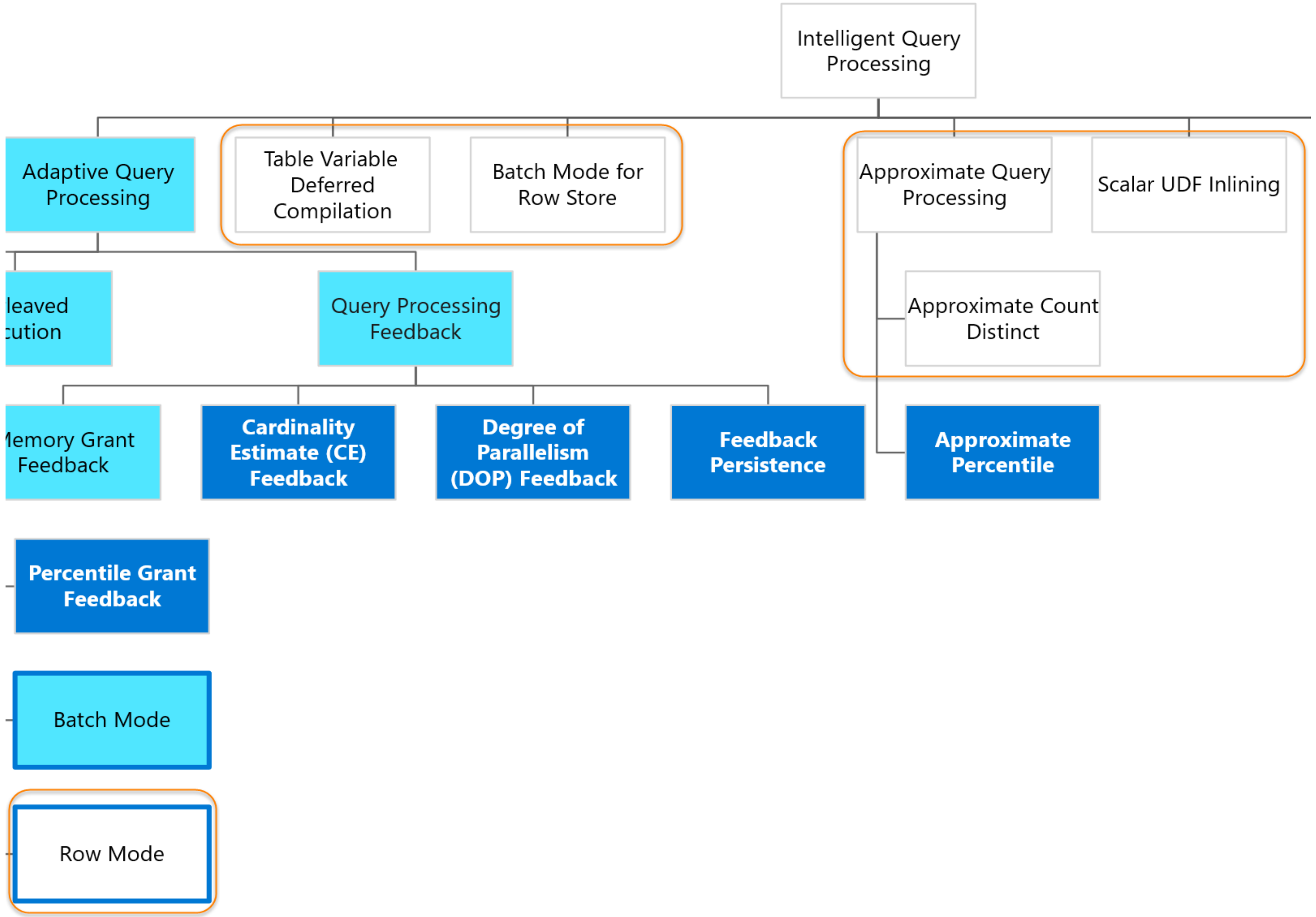
Execution	Memory Grant	Access Method	Algorithm
Parallel	In Memory	Seek	Join
Serial	Spill to Disk	Scan	Aggregate
		Seek + Scan	Sort

2017

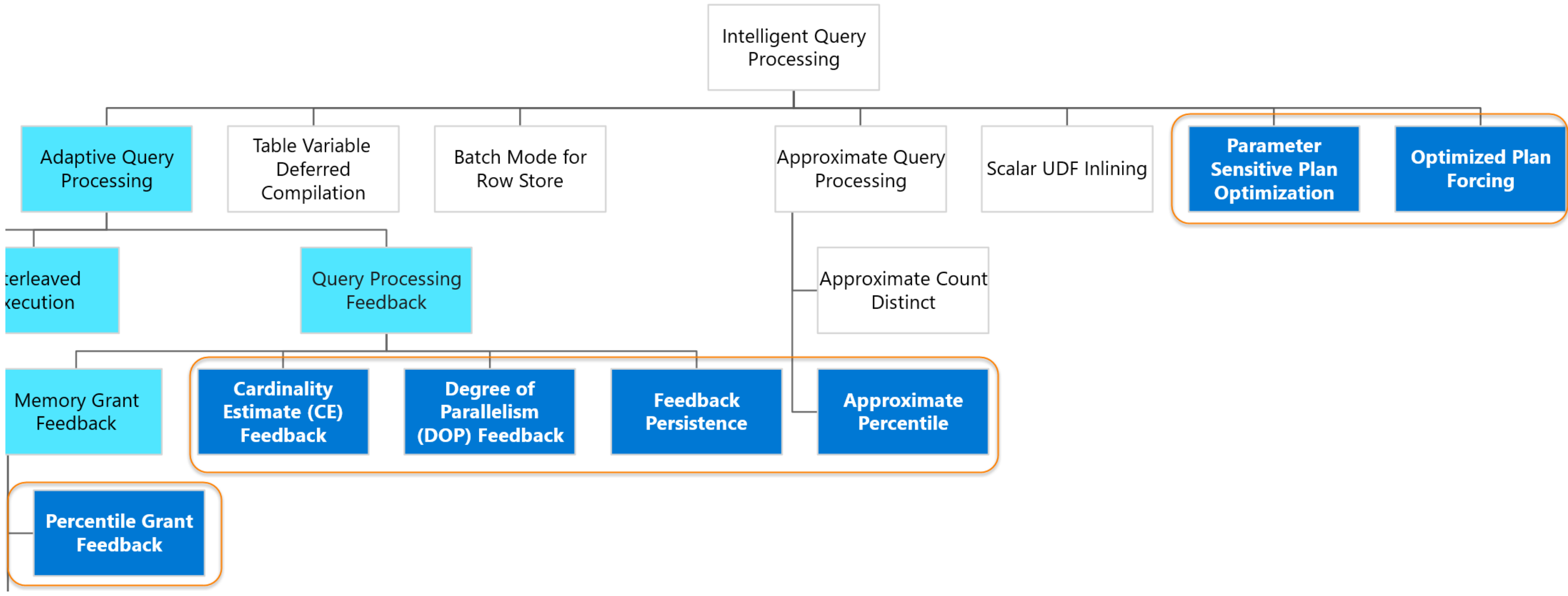

Intelligent Query Processing
Adaptive Query Processing
Table Variable Deferred Compilation
Batch Mode for Row Store
Approximate Processi
Adaptive Joins
Interleaved Execution
Query Processing Feedback
Approxi Di
Batch Mode
Memory Grant Feedback
Cardinality Estimate (CE) Feedback
Degree of Parallelism (DOP) Feedback
Feedback Persistence
Appr Per
Percentile Grant Feedback
Batch Mode

ON TOUR | NEW YORK
PASS

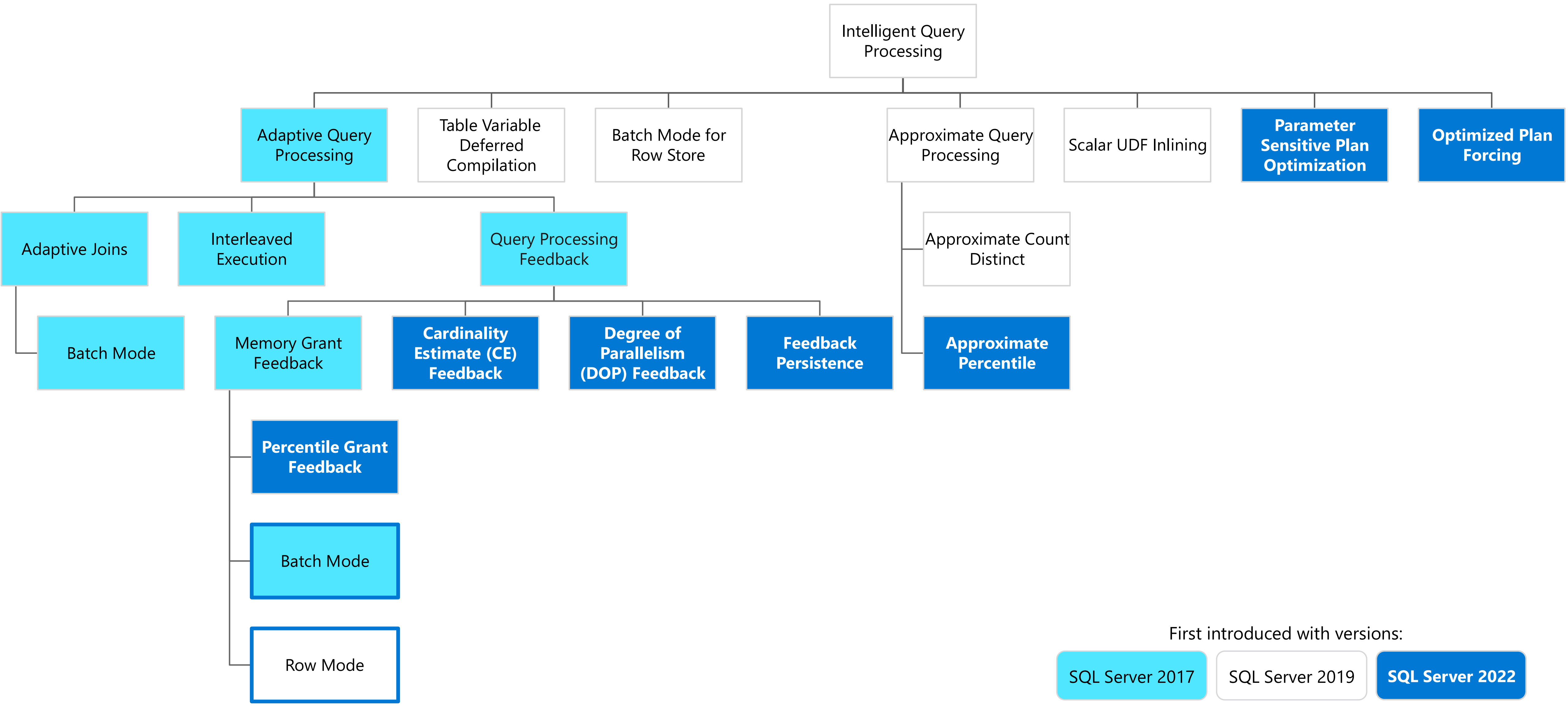
2019


Intelligent Query Processing
Adaptive Query Processing
Table Variable Deferred Compilation
Batch Mode for Row Store
Approximate Query Processing
Scalar UDF Inlining
leaved
cution
Query Processing Feedback
Approximate Count Distinct
lemory Grant Feedback
Cardinality Estimate (CE) Feedback
Degree of Parallelism (DOP) Feedback
Feedback Persistence
Approximate Percentile
Percentile Grant Feedback
Batch Mode
Row Mode

2022


Intelligent Query Processing
Adaptive Query Processing
Table Variable Deferred Compilation
Batch Mode for Row Store
Approximate Query Processing
Scalar UDF Inlining
Parameter Sensitive Plan Optimization
Optimized Plan Forcing
terleaved xecution
Query Processing Feedback
Approximate Count Distinct
Memory Grant Feedback
Cardinality Estimate (CE) Feedback
Degree of Parallelism (DOP) Feedback
Feedback Persistence
Approximate Percentile
Percentile Grant Feedback

ON TOUR | NEW YORK
PASS


Intelligent Query Processing
Adaptive Query Processing
Table Variable Deferred Compilation
Batch Mode for Row Store
Approximate Query Processing
Scalar UDF Inlining
Parameter Sensitive Plan Optimization
Optimized Plan Forcing
Adaptive Joins
Interleaved Execution
Query Processing Feedback
Approximate Count Distinct
Batch Mode
Memory Grant Feedback
Cardinality Estimate (CE) Feedback
Degree of Parallelism (DOP) Feedback
Feedback Persistence
Approximate Percentile
Percentile Grant Feedback
Batch Mode
Row Mode
First introduced with versions:
SQL Server 2017
SQL Server 2019
SQL Server 2022

Query Store Requirement

- Degree of parallelism (DOP) feedback
- Memory grant feedback (Percentile and Persistence mode)
- Optimized plan forcing with Query Store
- Parameter Sensitive Plan optimization (Not mandatory, but Recommended)

Thought process

- Code refactoring is expensive and time-consuming
- Heuristic-based, learn only from your dataset
- React to issues during compilation and execution
- Fix old limitations, such as scalar UDF inlining
- Learn via feedback
- Add intelligence to common operations, such as Approximate query processing

Principles

- Do no harm
- Improves the performance of existing workload with minimal implementation effort
- Available by default with the latest compatibility level
- Critical parallel workloads improve when running at scale, while remaining adaptive
- Options to disable

Compatibility Certification

- Predictable behavior after upgrades
- No need for recertifying T-SQL behavior
- Access to engine-level improvements (e.g., memory grants, adaptive joins, security features)
- Works across on-prem and cloud (Azure SQL)

I

ON TOUR | NEW YORK

SYS.DATABASE_SCOPED_CONFIGURATIONS

Disabling any of these features without changing the compatibility level

```
-- SQL Server 2017
ALTER DATABASE SCOPED CONFIGURATION SET
DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK = ON;

-- Starting with SQL Server 2019, and in Azure SQL Database
ALTER DATABASE SCOPED CONFIGURATION SET
BATCH_MODE_MEMORY_GRANT_FEEDBACK = OFF;
```

SYS.DM_EXEC_VALID_USE_HINTS

Disable any of these features for a specific query by using ‘USE HINT’ query hint

```
OPTION (USE HINT
('DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK'));

--The following example applies the hint to force
the legacy cardinality estimator to query_id 39,
identified in Query Store:
EXEC sys.sp_query_store_set_hints @query_id = 39,
@query_hints = N'OPTION (USE HINT
(''FORCE_LEGACY_CARDINALITY_ESTIMATION''))';
```

Which IQP features are Enterprise Edition only?

Problem

Solution

Caution

Further reading

ON TOUR | NEW YORK
PASS

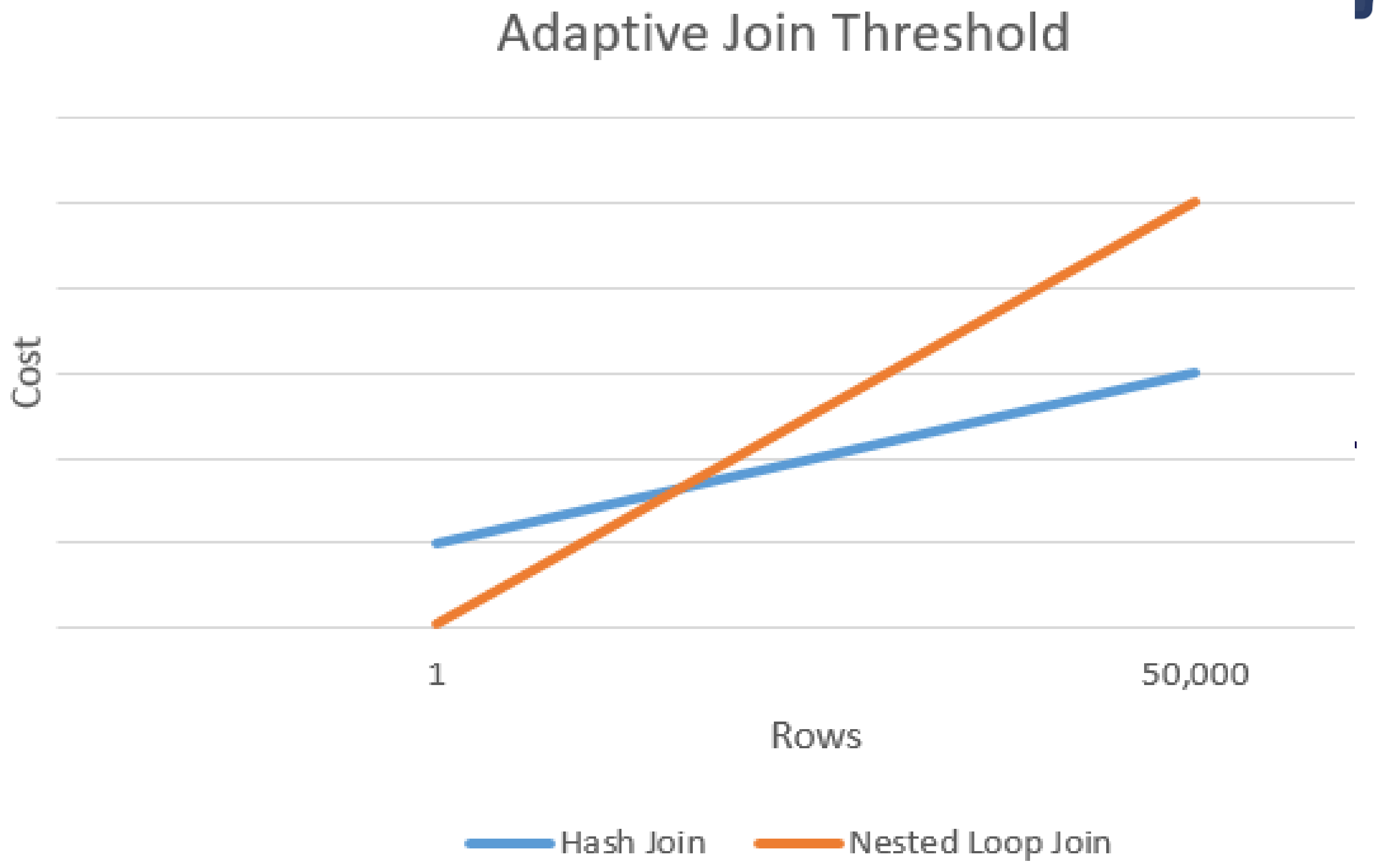
Adaptive Joins
Batch Mode

140+

- Join Hint
- Parameter-sensitive query

Adaptive Joins Batch Mode

140+



https://docs.microsoft.com/en-us/sql/relational-databases/performance/media/6_aqpjointhreshold.png?view=sql-server-ver15

Adaptive Joins Batch Mode

140+

- The query is a SELECT statement
- The join needs to be eligible with both Hash and Nested Loop joins
- Hash join uses batch mode
- Both joins should have the same outer reference
- Introduce a higher memory requirement

Adaptive Joins Batch Mode

140+

- Introducing Batch Mode Adaptive Joins
- Understand Adaptive Joins
- Hash Join or Nested Loops Join
- The Adaptive Join Threshold by Paul White
- A Little About Adaptive Joins In SQL Server by Erik Darling
- SQL Server 2017: How do Batch Mode Adaptive Joins work? By Erik Darling

Interleaved Execution MSTVFs

140+

- MSTVFs have a fixed cardinality guess of
  - 100 in SQL Server 2014 (12.x)
  - 1 in earlier versions

Interleaved Execution MSTVFs

140+

- Interleaved execution changes the unidirectional boundary between the optimization and execution phases
- Actual row counts are used to make better-informed decisions
- Greater performance impact with higher skew

Interleaved Execution MSTVFs

140+

- Must be read-only and NOT part of a data modification
- Must use a runtime constant
- Once cached, revised estimate is used for consecutive executions without re-instantiating the interleaved execution

Interleaved Execution MSTVFs

140+

- Introducing Interleaved Execution for Multi-Statement Table-Valued Functions
- Interleaved execution for MSTVFs feature in detail
- Multi-statement table-valued function (MSTVF)

Table Variable Deferred Compilation

150+

- Works ok with a low number of rows, but not as the number of rows increases
- Table variables do not have statistics
- Table variables do not have 'Automatic stats creation'
- Only inline index definitions
- Does not trigger recompile
- Fixed cardinality guess of 1

Table Variable Deferred Compilation

150+

- Optimizer delays the compilation
  - Same as what a temporary table does today
- Accurate cardinality – better execution plan
  - Example: Hash join instead of Nested loop join

Table Variable Deferred Compilation

150+

- Does not change any other characteristics
- Does not increase recompilation frequency
- Does not fix Parameter Sniffing issues
- Performance **may not be improved**

Table Variable Deferred Compilation

150+

- Public Preview of Table Variable Deferred Compilation in Azure SQL Database by Joe Sack
- Table variable deferred compilation
- Table Variable Deferred Compilation in SQL Server by Aaron Bertrand
- Demonstrating table variable deferred compilation by Joe Sack
- Improve Row Count Estimates for Table Variables without Changing Code by Greg Larsen

Memory Grant Feedback Batch Mode

140+

- Performance suffers from incorrect Memory Grant
  - Insufficient grant
  - Spill to disk
- Excessive grants
  - RESOURCE_SEMAPHORE waits
  - Wasted memory
  - Reduced concurrency

Memory Grant Feedback Batch Mode

140+

- Trigger recalculate
  - Result in a spill to disk
  - Granted memory > 2 x size of the actual used memory
- New SSMS property ‘IsMemoryGrantFeedbackAdjusted’ to track feedback

Memory Grant Feedback Batch Mode

140+

- Will disable itself for parameter-sensitive queries
- Grants under 1 MB will not be recalculated
- Changes are not captured in the Query Store with compatibility level 140
- Memory Grant honors the limitation by the resource governor or query hint

Memory Grant Feedback Batch Mode

140+

- Memory grant feedback
- Troubleshooting Variable Memory Grants in SQL Server by Erin Stellato
- Introducing Batch Mode Adaptive Memory Grant Feedback
- SQL Server 2017: How does Batch Mode Memory Grant Feedback Work? by Erik Darling
- Performance Demos of SQL's Intelligent Query Processing Feedback capabilities | Data Exposed

Memory Grant Feedback Row Mode

150+

- Row mode memory grant feedback expands on the batch mode
- You can track memory grant feedback events using the memory_grant_updated_by_feedback extended event.

Memory Grant Feedback Row Mode

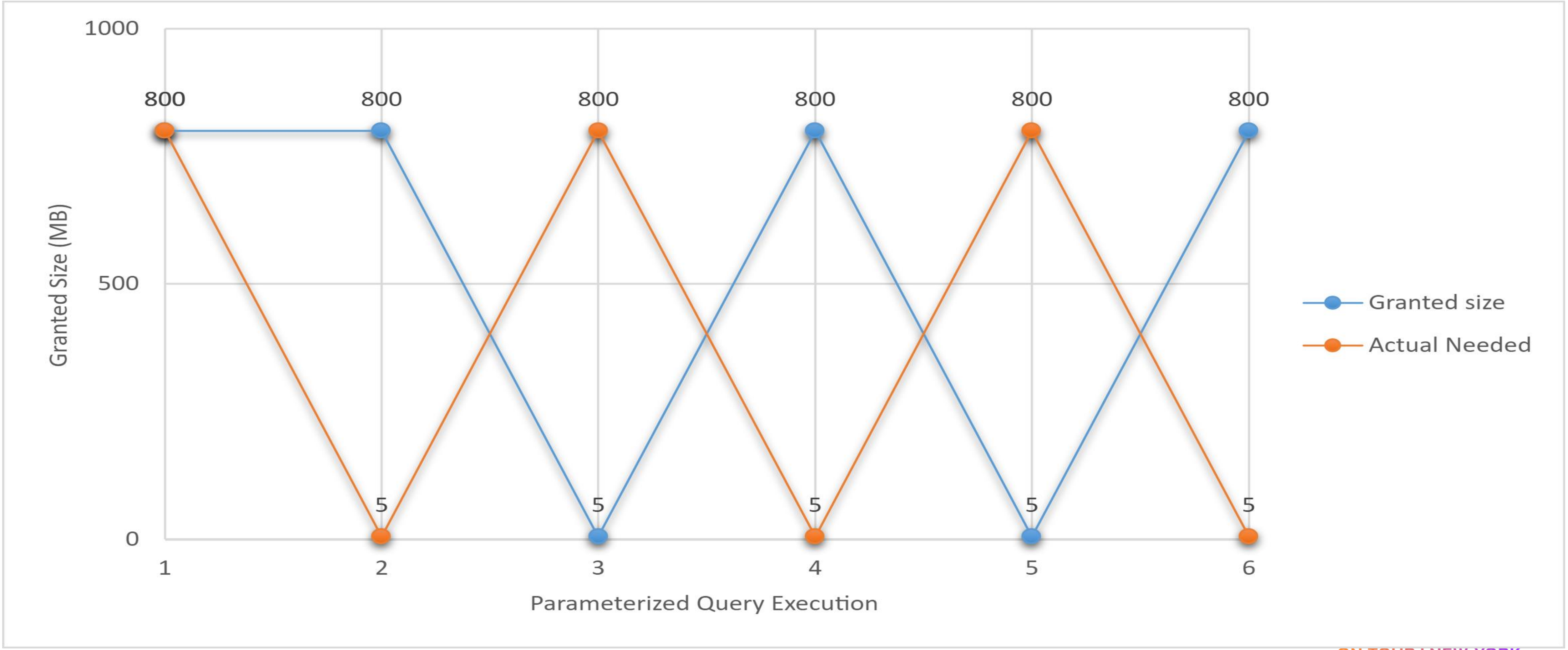
150+

- Public Preview of Row Mode Memory Grant Feedback in Azure SQL Database
- Row mode memory grant feedback
- What’s New in SQL Server 2019: Adaptive Memory Grants by Brent Ozar

Percentile grant Feedback

140+

- Grant size adjustments only accounted for the single most recently used grant
- This can trigger a severe anti-pattern of alternating request sizes and always-wrong memory grant adjustments
- Eventually disabling the memory grant feedback feature



https://www.microsoft.com/en-us/sql-server/blog/2022/10/27/memory-grant-feedback-persistence-and-percentile-grant/

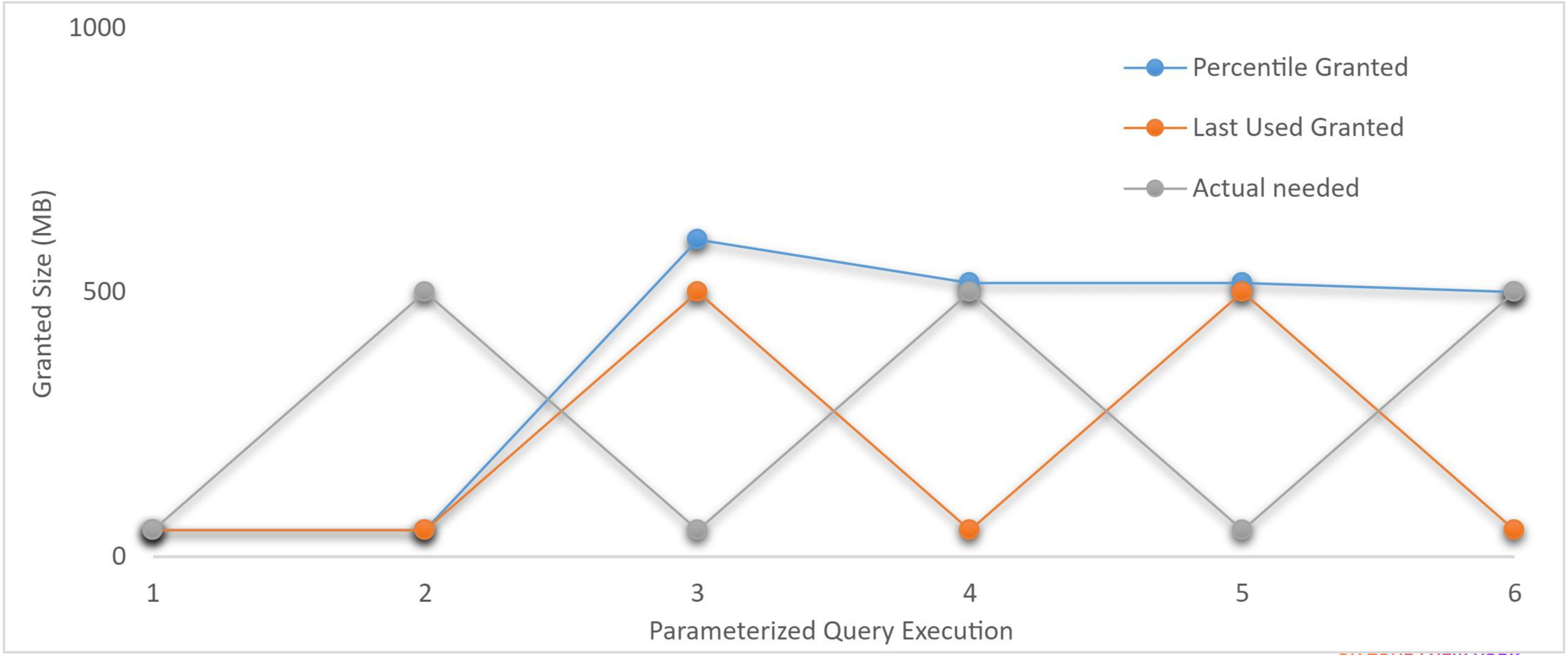
ON TOUR | NEW YORK

PASS

Percentile grant Feedback

140+

- Using a percentile-based calculation over the recent history of the query based on a larger set of data points
- Always err toward providing more memory to avoid spills



https://www.microsoft.com/en-us/sql-server/blog/2022/10/27/memory-grant-feedback-persistence-and-percentile-grant/

Percentile grant Feedback

140+

- This feature was introduced in SQL Server 2022 (16.x), but is available with CE 140+
- Has no effect if Query Store is not enabled in a "read write‟ state

Percentile grant Feedback

140+

- Percentile and persistence mode memory grant feedback
- Memory Grant Feedback: Persistence and Percentile Grant By Kate Smith
- Azure SQL and SQL Server 2022: Intelligent Database Futures by Pedro Lopes

Cardinality Estimate (CE) Feedback

160+

- No single set of CE models and assumptions can accommodate the vast array of customer workloads and data distributions
- Addresses perceived regression issues resulting from incorrect CE model assumptions when using the default CE
- The scenarios include Correlation, Join Containment, and Optimizer row goal

Cardinality Estimate (CE) Feedback

160+

- CE feedback identifies model-related assumptions and evaluates whether they're accurate for repeating queries
- If it looks incorrect, a subsequent execution is tested with a query plan that adjusts the impactful CE model assumption and verifies if it helps
- If it improves plan quality, the old query plan is replaced with a query plan that uses the appropriate USE HINT query hint that adjusts the estimation model, implemented through the Query Store hint mechanism.

Cardinality Estimate (CE) Feedback

160+

- Even though Query store for Secondary replica is enabled in SQL 2022: CE feedback isn't replica-aware
- If a query uses hard-coded query hints or uses Query Store hints set by the user, CE feedback won't be used for that query
- SQL 2022 CU8 introduced a bug related to CE feedback, but it was resolved in CU12

Cardinality Estimate (CE) Feedback

160+

- Cardinality estimation (CE) feedback
- Cardinality Estimation: A Comprehensive Look by Kate Smith
- Performance Demos of SQL's Intelligent Query Processing Feedback capabilities | Data Exposed
- SQL Server 2022: Cardinality Estimation Feedback by Erik Darling
- A Little About Cardinality Estimation Feedback In SQL Server 2022 by Erik Darling

Degree of Parallelism (DOP) Feedback

160+

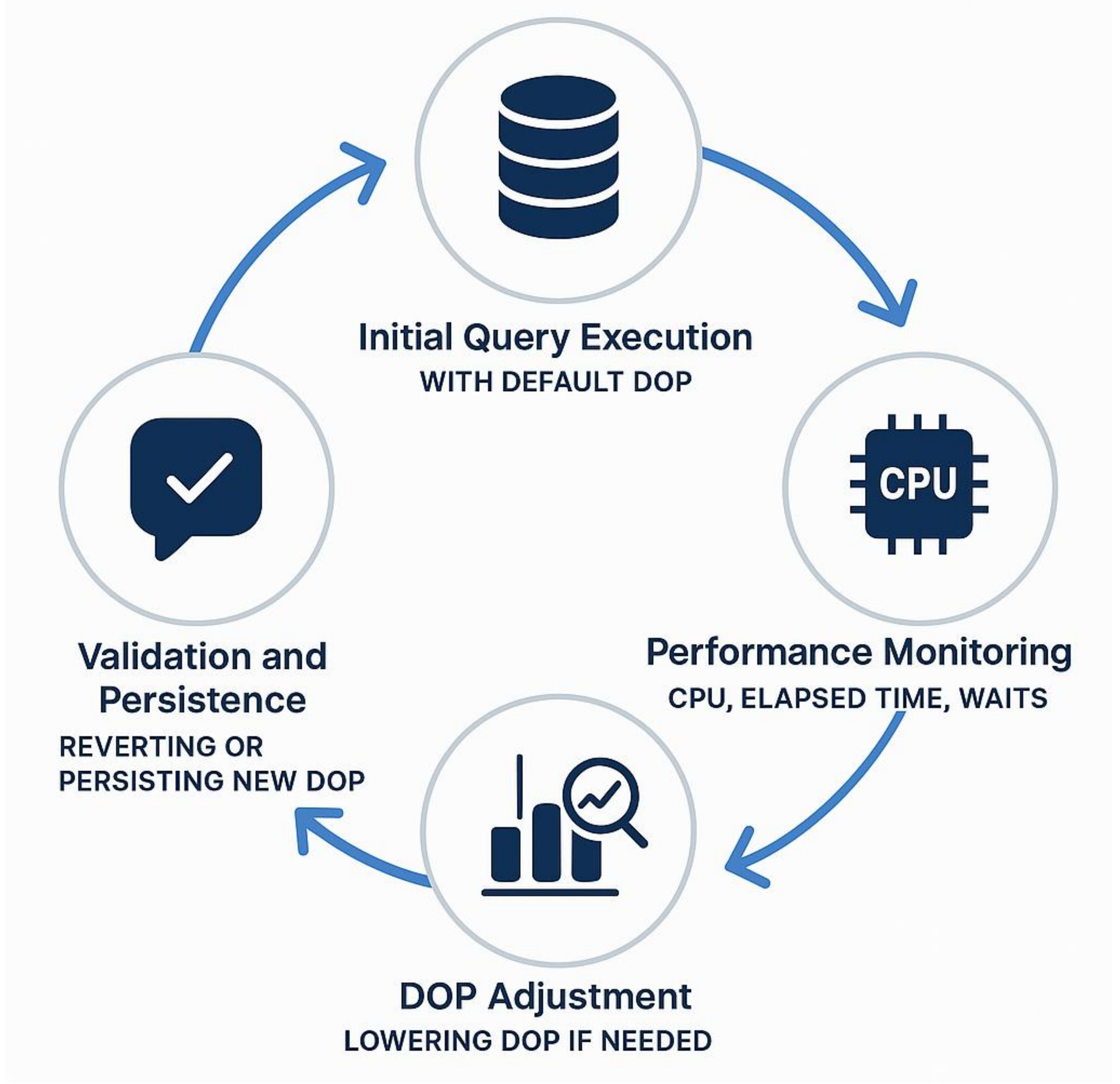
- Addresses suboptimal usage of parallelism for repeating queries by identifying parallelism inefficiencies
- Instead of incurring the pains of an all-encompassing default or manual adjustments to each query, DOP feedback self-adjusts DOP
- OLTP-centric queries that are executed in parallel could experience performance issues when the time spent coordinating all threads outweighs the advantages of using a parallel plan

Degree of Parallelism (DOP) Feedback

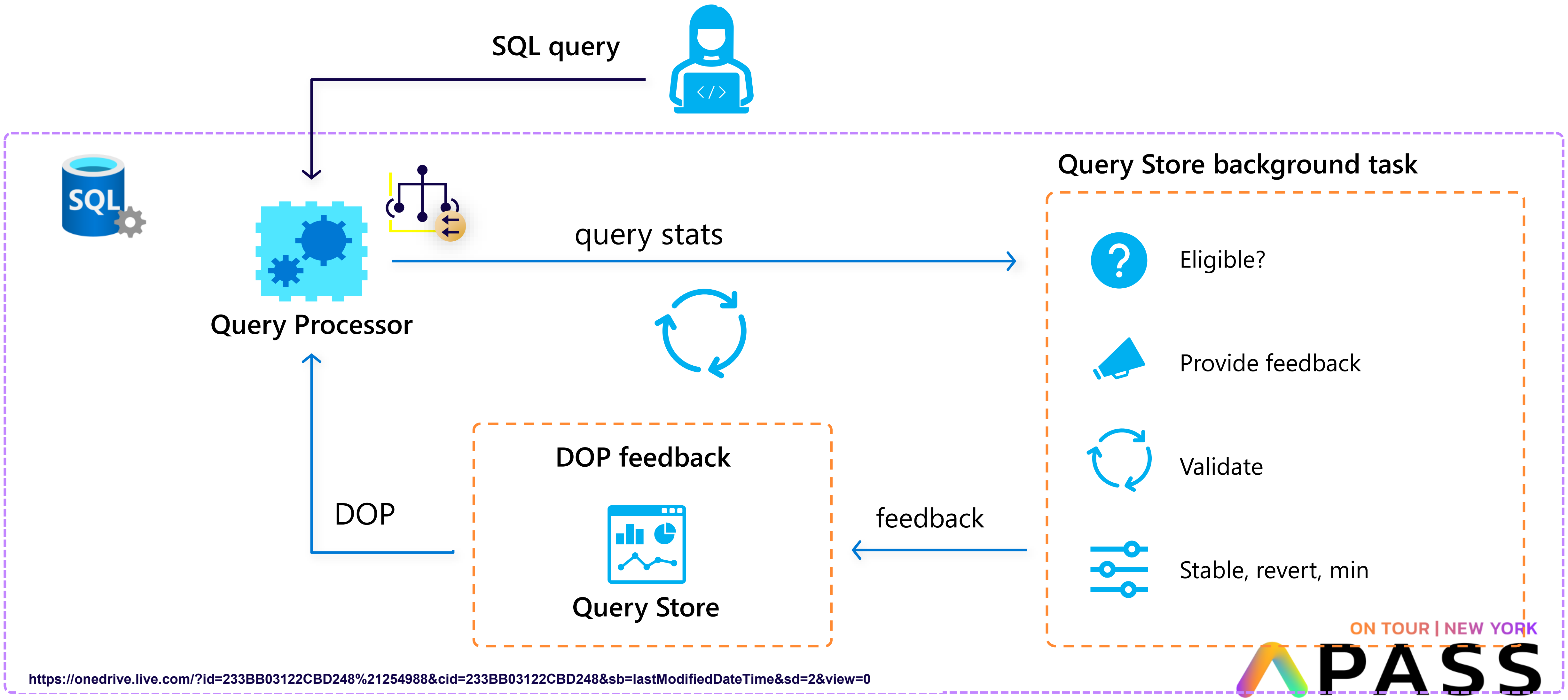
160+

- Parallelism inefficiencies for repeating queries based on elapsed time and waits
- If parallelism usage is deemed inefficient, DOP feedback will lower the DOP for the subsequent execution of the query
- Minimum DOP for any query adjusted with DOP feedback is 2

Visualizing the Feedback Loop



DOP feedback architecture



https://onedrive.live.com/?id=233BB03122CBD248%21254988&cid=233BB03122CBD248&sb=lastModifiedDateTime&sd=2&view=0

Degree of Parallelism (DOP) Feedback

160+

- To enable DOP feedback, enable the DOP_FEEDBACK database scoped configuration in a database
- The Query Store must be enabled for every database where DOP feedback is used, and in the "Read write" state
- Stable feedback is reverified upon plan recompilation and may readjust up or down, but never above MAXDOP setting (including a MAXDOP hint)
- DOP feedback is Replica aware

Degree of Parallelism (DOP) Feedback

160+

- Degree of parallelism (DOP) feedback
- Intelligent Query Processing: degree of parallelism feedback by Kate Smith
- SQL Server 2022: Built-in Query Intelligence (Ep. 3) | Data Exposed
- Performance Demos of SQL's Intelligent Query Processing Feedback capabilities | Data Exposed
- What's The Point Of DOP Feedback In SQL Server 2022? by Erik Darling

Feedback Persistence

160+

- The existing feature does not work with the plan eviction
- Poor performance the first few times a query is executed after an eviction
- Provides new functionality to
  - Memory grant feedback (140+)
  - Cardinality Estimate Feedback
  - degree of parallelism (DOP) feedback

Feedback Persistence

160+

- The Query Store must be enabled in read-write mode for every database where the persistence portion of this feature is used
- Only verified feedback is stored in query store

Feedback Persistence

160+

- Query Store to be enabled for the database and in a “read-write" state
- No impact if Query Store is not enabled
- During failover, the memory grant feedback from the old primary replica is applied to the new primary replica

Feedback Persistence

160+

- Percentile and persistence mode memory grant feedback
- Persistence for cardinality estimation (CE) feedback
- Persistence for degree of parallelism (DOP) feedback

Batch Mode on rowstore

150+

- Row by Row processing is slow and CPU-intensive
- Columnstore indexes may not be appropriate for some applications
- Features might restrict the use of the Columnstore index
  - Trigger
  - Cursor
  - Persisted computed columns

Batch Mode on rowstore

150+

- Uses heuristics – during the estimation phase
  - Table sizes
  - Operators used
  - Estimated cardinalities
- Additional checkpoints, to evaluate plans with batch mode
- Support for all existing batch mode-enabled operators
- Workload consists of analytics queries, especially with joins or aggregates
- Workload that is CPU-bound

Batch Mode on rowstore

150+

- Batch mode restriction is always applicable
  - Example Queries involving cursors
- Not applicable for in-memory OLTP tables
- Not applicable for any index other than on-disk heaps and B-trees
- Won't kick in for
  - Large Object (LOB) column
  - XML column
  - Sparse column sets
- Two features are independent

Batch Mode on rowstore

150+

- Introducing Batch Mode on Rowstore
- Batch mode on rowstore
- Workloads that might benefit from batch mode on rowstore

ON TOUR | NEW YORK PASS

Approximate Count Distinct

2019

- Responsiveness is important than absolute precision
- Example
  - Dashboard scenarios
  - Data science is trying to understand data distributions

Approximate Count Distinct

2019

- Access to data sets that are millions of rows or higher
- Aggregation of a column or columns that have many distinct values
- Uses less memory compared to exhaustive COUNT DISTINCT
- Based on the HyperLogLog algorithm

Approximate Count Distinct

2019

- The function implementation guarantees up to a 2% error rate within a 97% probability
- This feature is available starting with SQL Server 2019 (15.x), regardless of the compatibility level

Approximate Count Distinct

2019

- Approximate query processing
- APPROX_COUNT_DISTINCT (Transact-SQL)
- SQL Server 2019 APPROX_COUNT_DISTINCT Function by Aaron Bertrand

Approximate Percentile

2022

- Large datasets where negligible error with a faster response is acceptable as compared to an accurate percentile value with a slow response

Approximate Percentile

2022

- Approximate percentile aggregate functions compute percentiles for a large dataset with acceptable rank-based error bounds to help make rapid decisions
- Approximate percentile functions use KLL sketch. The sketch is built by reading the stream of data
- These functions provide rank-based error guarantees, not value-based

Approximate Percentile

2022

- The output of the function may not be the same in all executions, since it uses a randomized algorithm
- The function implementation guarantees up to a 1.33% error bound within a 99% confidence level

Approximate Percentile

2019

- Approximate query processing
- APPROX_PERCENTILE_DISC (Transact-SQL)
- APPROX_PERCENTILE_CONT (Transact-SQL)
- Additional T-SQL Improvements in SQL Server 2022 by Itzik Ben-Gan

Scalar UDF Inlining

150+

- Iterative invocation
- Lack of costing
- Interpreted execution
- Serial Execution
- Imperative code does not scale

Scalar UDF Inlining

150+

- UDFs are automatically transformed into
  - Scalar Expressions
  - Scalar subqueries
- Further optimization followed by transformation
- Refactors the Imperative code into Relational algebraic expression – Froid Framework
- Resulting execution plan
  - Efficient
  - Set-Oriented
  - Parallel
- New SSMS property ‘ContainsInlineScalarTsqlUdfs’ to track inlining

Scalar UDF Inlining

150+

```
-- Transact-SQL Function Clauses
<function_option>::=
{
      [ ENCRYPTION ]
    | [ SCHEMABINDING ]
    | [ RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT
    | [ EXECUTE_AS_Clause ]
    | [ INLINE = { ON | OFF }]
}
```

ON TOUR | NEW YORK
PASS

Scalar UDF Inlining

150+

- Scalar UDF inlining
- KB4538581 - FIX: Scalar UDF Inlining issues in SQL Server 2022 and 2019
- Get Your Scalar UDFs to Run Faster Without Code Changes by Greg Larsen
- Rewriting T-SQL Scalar UDFs So They're Eligible For Automatic Inlining In SQL Server by Erik Darling
- Another Trick For Working Around Scalar UDF Inlining Restrictions In SQL Server by Erik Darling
- Finding Froid’s Limits: Testing Inlined User-Defined Functions by Brent Ozar

Scalar UDF Inlining

150+

- UDF Inlining Demos by Erik Darling
- Rewriting Scalar UDFs As Inline Table Valued Functions With CTEs by Erik Darling
- Scalar UDF Inlining in SQL Server 2019 by Aaron Bertrand

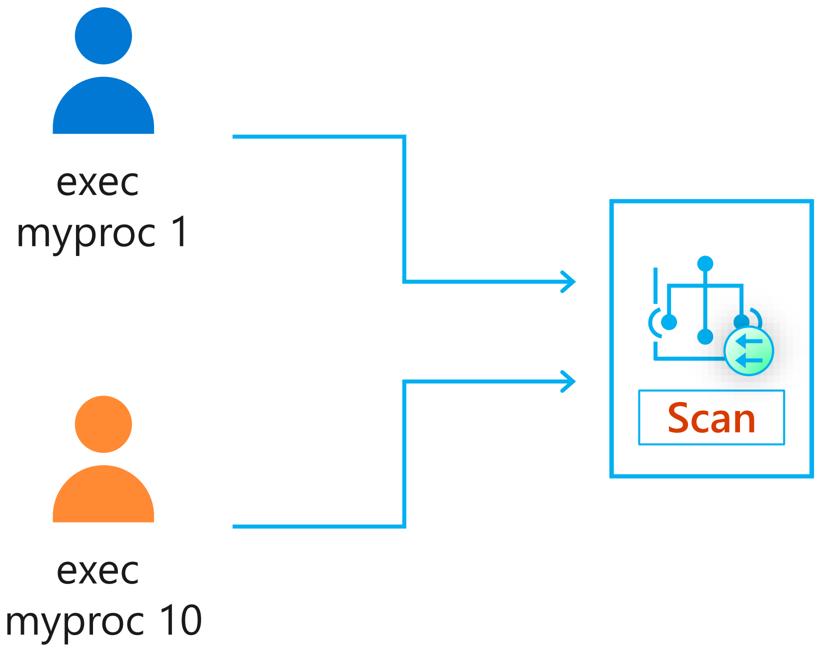
Parameter Sensitive Plan Optimization

160+

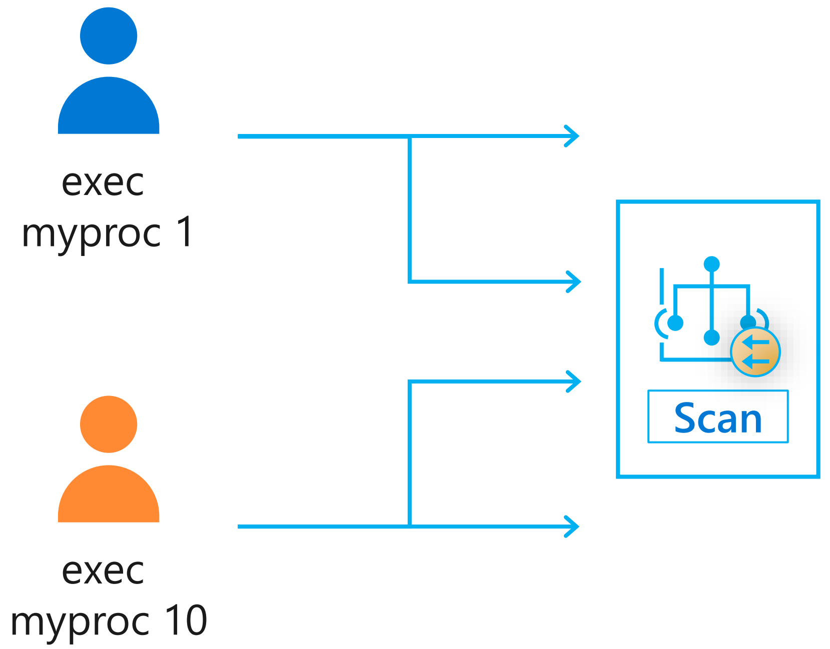
- A single cached plan for a parameterized query isn't optimal for all possible incoming parameter values
- PSP optimization automatically enables multiple, active cached plans for a single parameterized statement

Parameter Sensitive Plan Optimization

Before



With PSP optimization



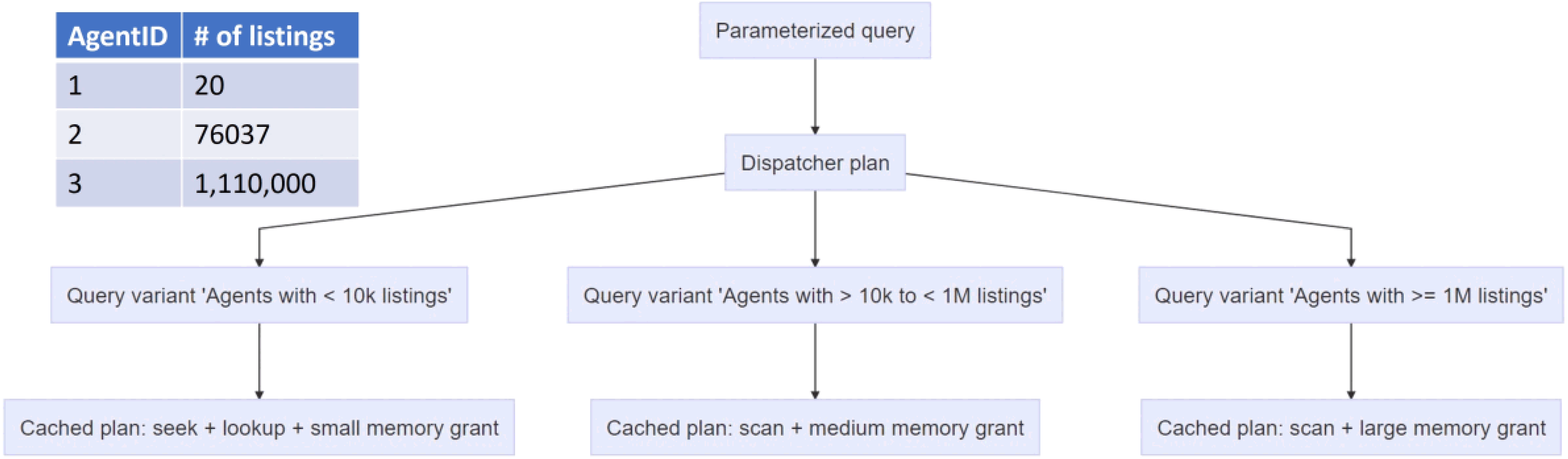
https://onedrive.live.com/?id=233BB03122CBD248%21254988&cid=233BB03122CBD248&sb=lastModifiedDateTime&sd=2&view=0

Parameter Sensitive Plan Optimization

160+

- During the initial compilation, column statistics histograms identify non-uniform distributions and evaluate the most at-risk parameterized predicates, up to three out of all available predicates
- PSP feature limits the number of predicates that are evaluated to avoid bloating the plan cache and the Query Store
- Initial compilation produces a dispatcher plan that contains the PSP optimization logic called a dispatcher expression
- A dispatcher plan maps to query variants based on the cardinality range boundary values and predicates

AgentID	# of listings
1	20
2	76037
3	1,110,000


Parameterized query
Dispatcher plan
Query variant 'Agents with < 10k listings'
Query variant 'Agents with > 10k to < 1M listings'
Query variant 'Agents with >= 1M listings'
Cached plan: seek + lookup + small memory grant
Cached plan: scan + medium memory grant
Cached plan: scan + large memory grant

https://learn.microsoft.com/en-us/sql/relational-databases/performance/parameter-sensitive-plan-optimization?view=sql-server-ver17

Parameter Sensitive Plan Optimization

160+

- The PSP optimization feature currently only works with equality predicates (Major change in 2025)
- Query variant plans will recompile independently as needed, as with any other query plan type
- When multiple predicates are part of the same table, PSP optimization will select the predicate that has the most data skew based on the underlying statistics histogram

Parameter Sensitive Plan Optimization

160+

- Parameter Sensitive Plan optimization
- Parameter sensitivity
- Parameters and execution plan reuse
- Parameter Sensitive Plan Optimization in SQL 2022 … As Cool as it Sounds? by Erin Stellato and Hugo Kornelis
- SQL Server 2022: Built-in Query Intelligence (Ep. 3) | Data Exposed
- PSPO: How SQL Server 2022 Tries to Fix Parameter Sniffing by Brent Ozar

Optimized Plan Forcing

2022

- Optimized plan forcing reduces compilation overhead for repeating forced queries

Optimized Plan Forcing

2022

- During the compilation process, a threshold based on estimating the time spent in optimization (based on the query optimizer input tree) will determine whether an optimization replay script is created
- These runtime metrics include the number of objects accessed, joins, optimization tasks executed during optimization, and the actual optimization time.

Optimized Plan Forcing

2022

- Only query plans that go through full optimization are eligible
- Statements with the RECOMPILE hint and distributed queries are not eligible
- Even if an optimization replay script was generated, it might not be persisted in the Query Store if the Query Store configured capture policies criteria aren’t met. For example: number of executions of that statement and its cumulated compile and execution times

Optimized Plan Forcing

2022

- Optimized plan forcing with Query Store
- Stabilizing Performance with Query Store by Erin Stellato

Query Store Hints

160+

- Need immediate behavior change
- No access to source code
- Plan guide – never easy to use
- Example
  - Recompile a query on each execution.
  - Cap the memory grant size for a bulk insert operation.
  - Limit the maximum degree of parallelism for a statistics update operation.
  - 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.
  - Disable row goal optimization for a SELECT TOP query.

Query Store Hints

160+

Query Executed

Query captured in Query Store

DBA creates a Query Store hint on a query

Query executes using Query Store hint

Query Store Hints

160+

- Query store hints override statement-level hints (hard-coded) and plan guide hints
- If hints contradict, query execution will not be blocked
- Query Store hints are persisted and survive restarts and failovers

Query Store Hints

160+

- Query Store hints
- Query Store hints best practice
- Supported query hints
- Query Store Hints in Azure SQL Database | Data Exposed
- Query Store Performance Overhead…Updated by Erin Stellato

Demo

Further Reading

- Intelligent query processing in SQL databases
- Editions and supported features of SQL Server 2022
- Intelligent query processing demo - I
- Intelligent query processing demo -II
- Compatibility certification
- Batch Mode Bitmaps in SQL Server by Paul White

Thank you

Reach out to me with questions/comments.
You are guaranteed an answer!

Taiob Ali

@sqlworldwide

Taiob at sqlworldwide dot com

https://sqlworldwide.com/