



# Introduction to Intelligent Query Processing

John Deardurff



# What is in the session?

**A History of Intelligent Query Processing?**

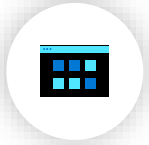
**Enabling and Disabling Features**

**Adaptive Query Processing (2017)**

**Intelligent Query Processing (2019)**

**New Features of IQP (2022)**

# Learn more



Learn more about SQL Server 2022

[aka.ms/sqlserver2022](https://aka.ms/sqlserver2022)



Watch a technical deep-dive on SQL Server 2022

[aka.ms/sqlserver2022mechanics](https://aka.ms/sqlserver2022mechanics)



Don't miss out on Data Exposed (Look for SQL Server 2022 – Episode 3)

[aka.ms/dataexposed](https://aka.ms/dataexposed)



More information on Intelligent Query Processing  
(Currently does not have 2022 information.)

[aka.ms/IQP](https://aka.ms/IQP)

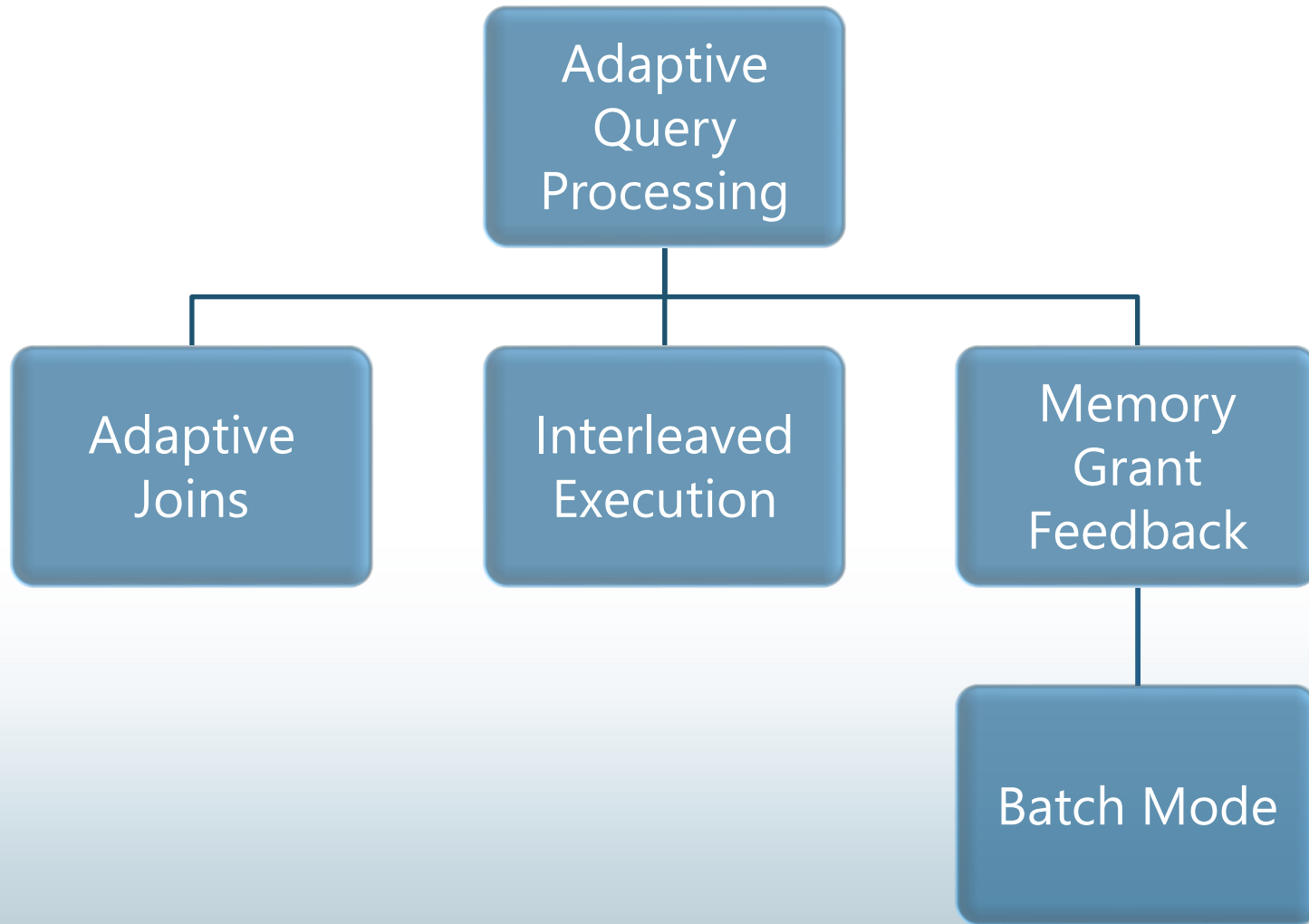
<https://aka.ms/IQPDemos>

<https://aka.ms/SQLCE>

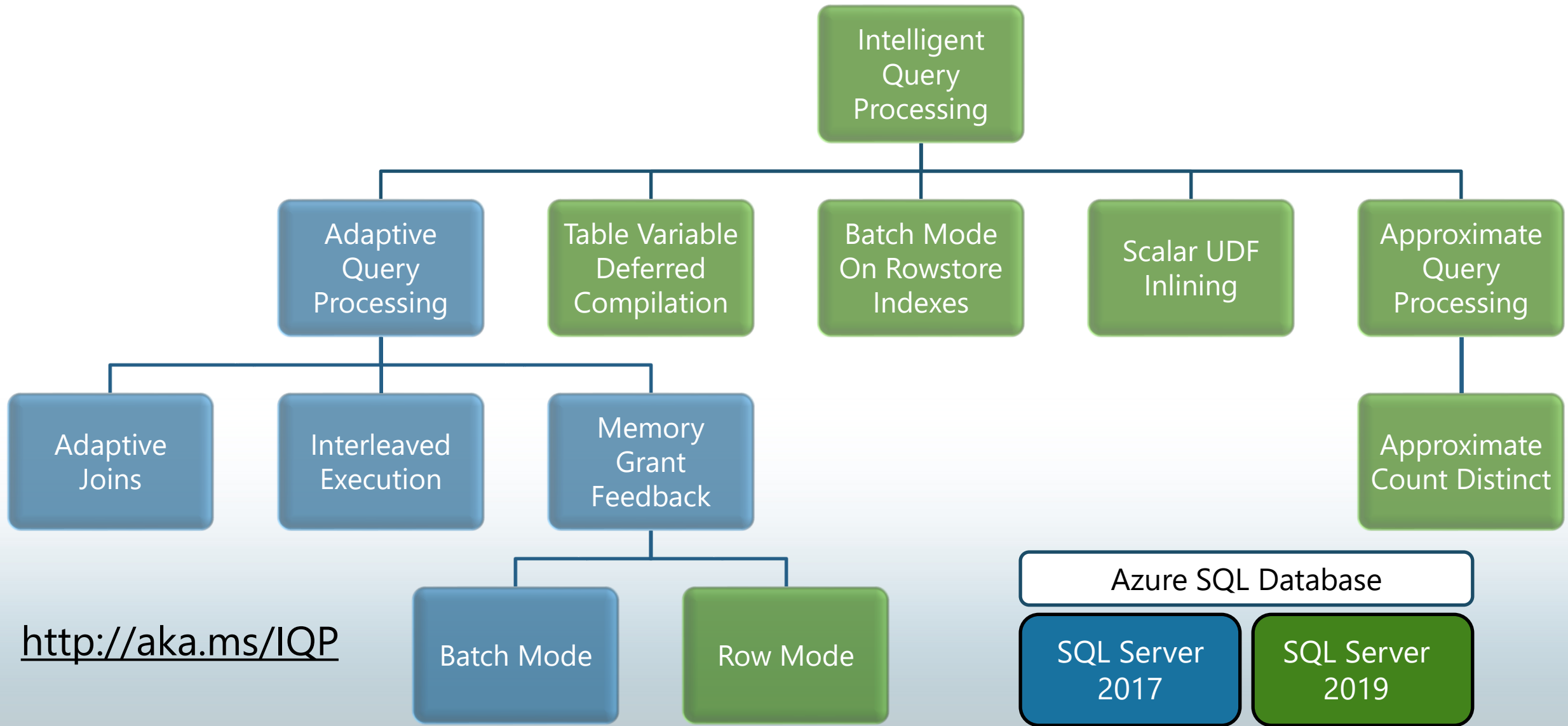
# In a World... before IQP



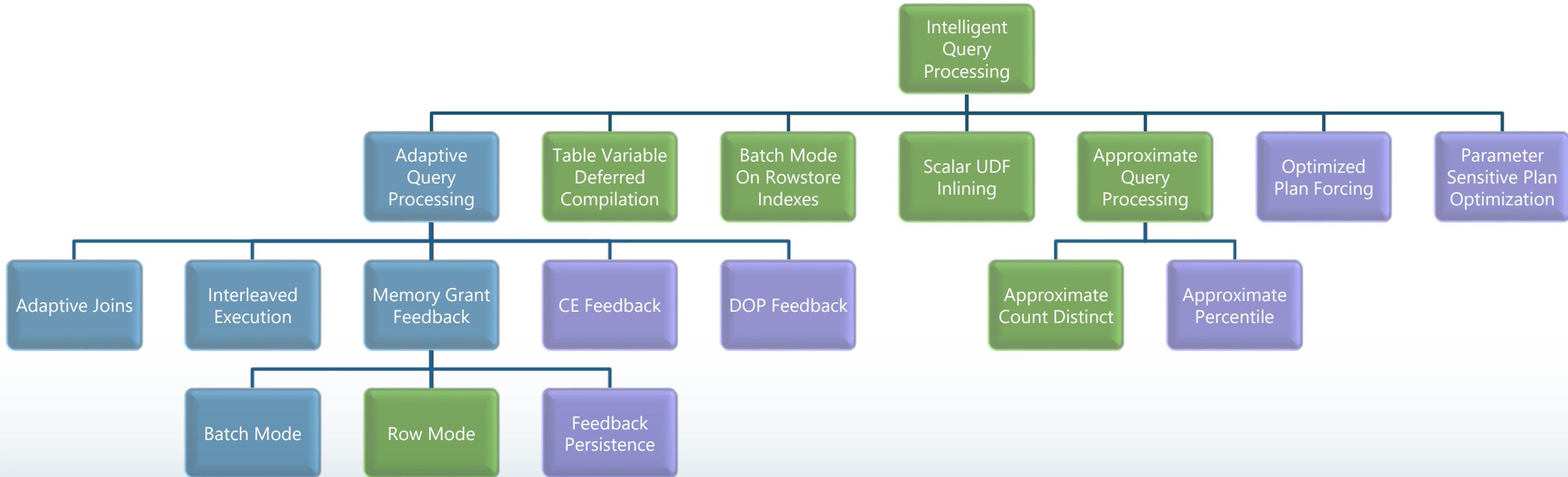
# Adaptive Query Processing (2017)



# Intelligent Query Processing (2019)



# Intelligent Query Processing (2022)



Azure SQL Database

2017

2019

2022

# Enabling and Disabling – Instance Level

## For SQL Server 2017 Features

- Enabled by default in Compatibility level 140 or higher
- To disable change compatibility level to 130 or lower

## For SQL Server 2019 Features

- Enabled by default in Compatibility level 150 or higher
- To disable change compatibility level to 140 or lower

## For SQL Server 2022 Features

- Enabled by default in Compatibility level 160 or higher
- To disable change compatibility level to 150 or lower



# Enabling and Disabling – Database Level

Different settings for 2017 vs Azure SQL, SQL Server 2019 and higher

```
ALTER DATABASE SCOPED CONFIGURATION SET DISABLE_BATCH_MODE_ADAPTIVE_JOINS = ON|OFF;
```

```
ALTER DATABASE SCOPED CONFIGURATION SET BATCH_MODE_ADAPTIVE_JOINS = ON|OFF;
```

To get a list of Database Scoped Configuration settings

```
SELECT * From sys.database_scoped_configurations;
```

configuration_id	name	value
7	INTERLEAVED_EXECUTION_TVF	1
8	BATCH_MODE_MEMORY_GRANT_FEEDBACK	1
9	BATCH_MODE_ADAPTIVE_JOINS	1
10	TSQL_SCALAR_UDF_INLINING	1
16	ROW_MODE_MEMORY_GRANT_FEEDBACK	1
18	BATCH_MODE_ON_ROWSTORE	1
19	DEFERRED_COMPILATION_TV	1
28	PARAMETER_SENSITIVE_PLAN_OPTIMIZATION	1
31	CE_FEEDBACK	1
33	MEMORY_GRANT_FEEDBACK_PERSISTENCE	1
34	MEMORY_GRANT_FEEDBACK_PERCENTILE_GRANT	1
35	OPTIMIZED_PLAN_FORCING	0

# Enabling and Disabling – Statement Level

You can disable features at the statement scope if necessary.

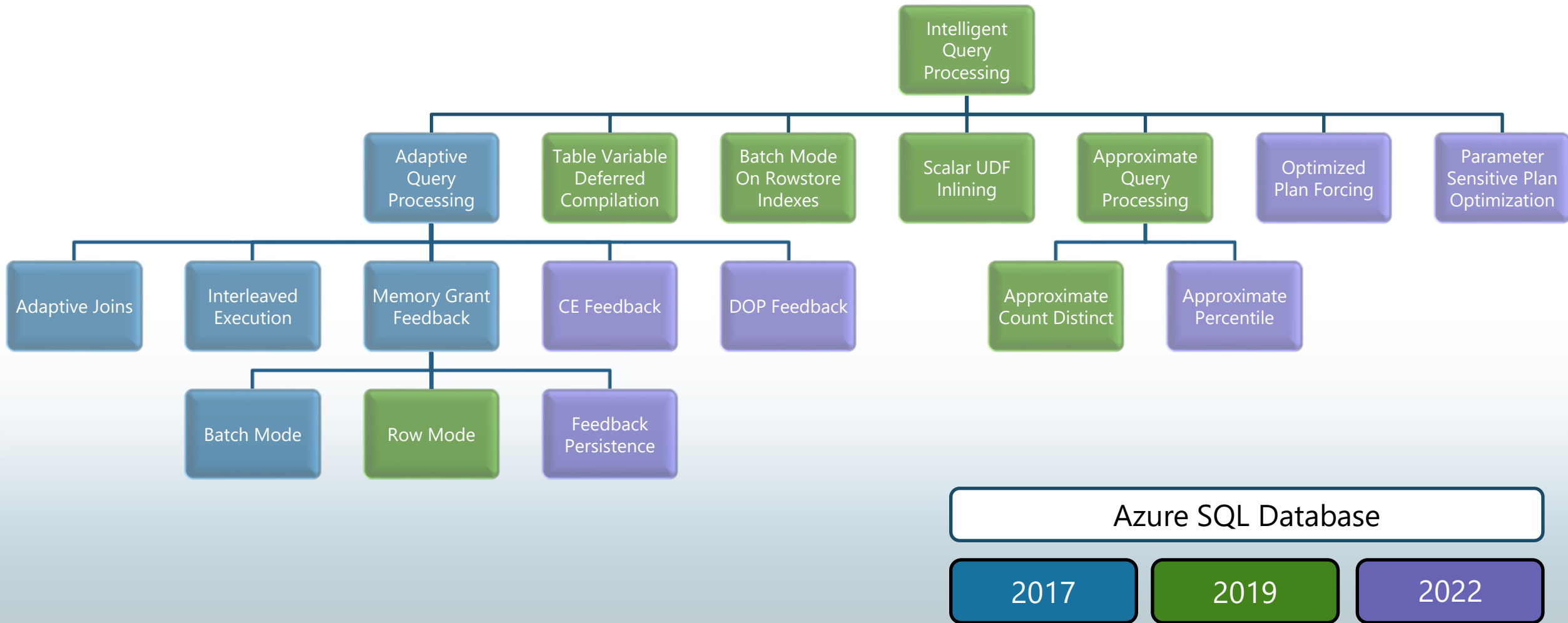
```
<statement>  
OPTION (USE HINT('DISABLE_BATCH_MODE_ADAPTIVE_JOINS'));
```

To get a list of valid query use hints

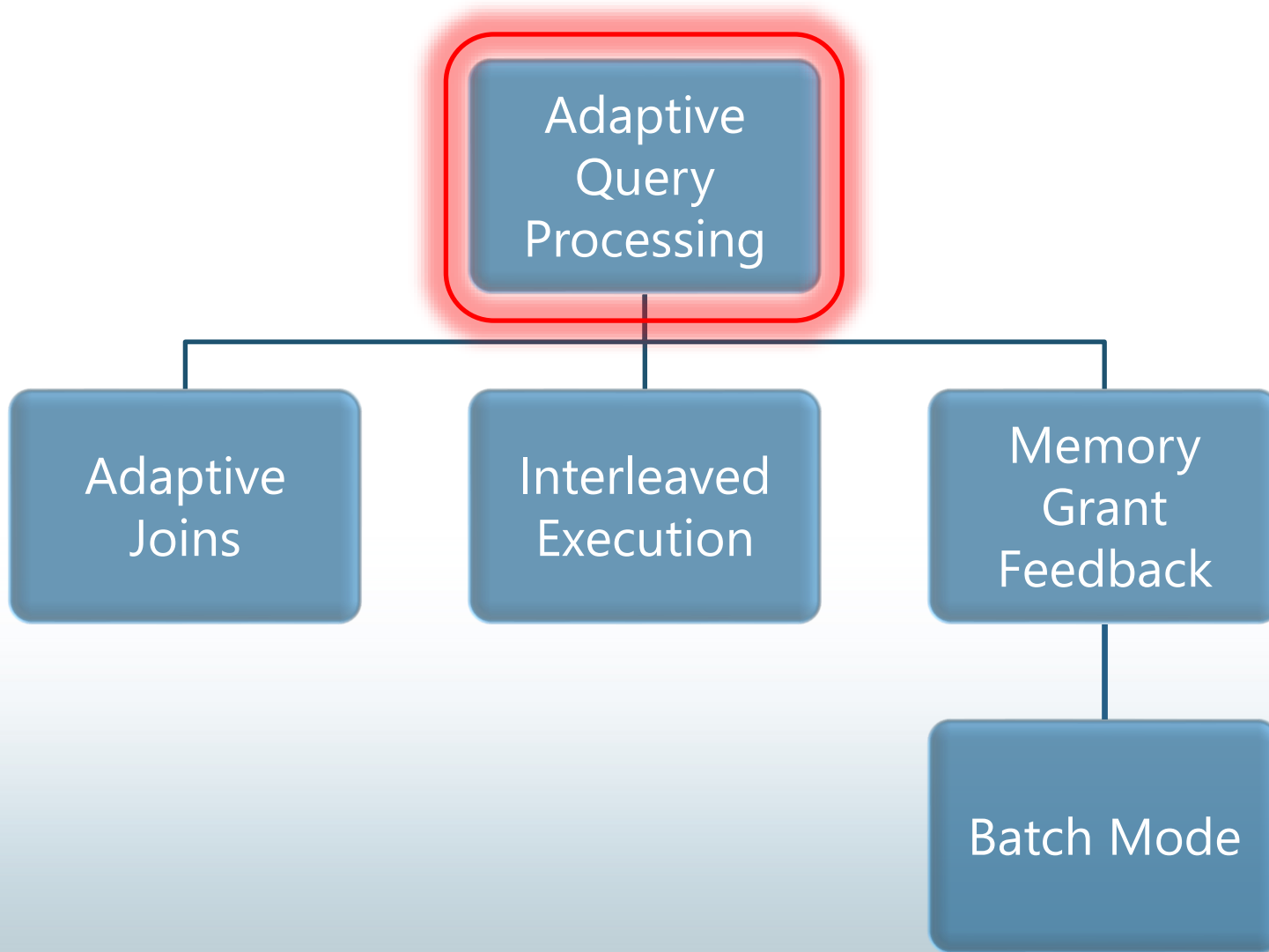
```
SELECT * FROM sys.dm_exec_valid_use_hints;
```

name
DISABLE_INTERLEAVED_EXECUTION_TVF
DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK
DISABLE_BATCH_MODE_ADAPTIVE_JOINS
DISABLE_ROW_MODE_MEMORY_GRANT_FEEDBACK
DISABLE_DEFERRED_COMPILATION_TV
DISABLE_TSQL_SCALAR_UDF_INLINING
ASSUME_FULL_INDEPENDENCE_FOR_FILTER_ESTIMATES
ASSUME_PARTIAL_CORRELATION_FOR_FILTER_ESTIMATES
DISABLE_CE_FEEDBACK
DISABLE_MEMORY_GRANT_FEEDBACK_PERSISTENCE
DISABLE_DOP_FEEDBACK
DISABLE_OPTIMIZED_PLAN_FORCING

# Intelligent Query Processing



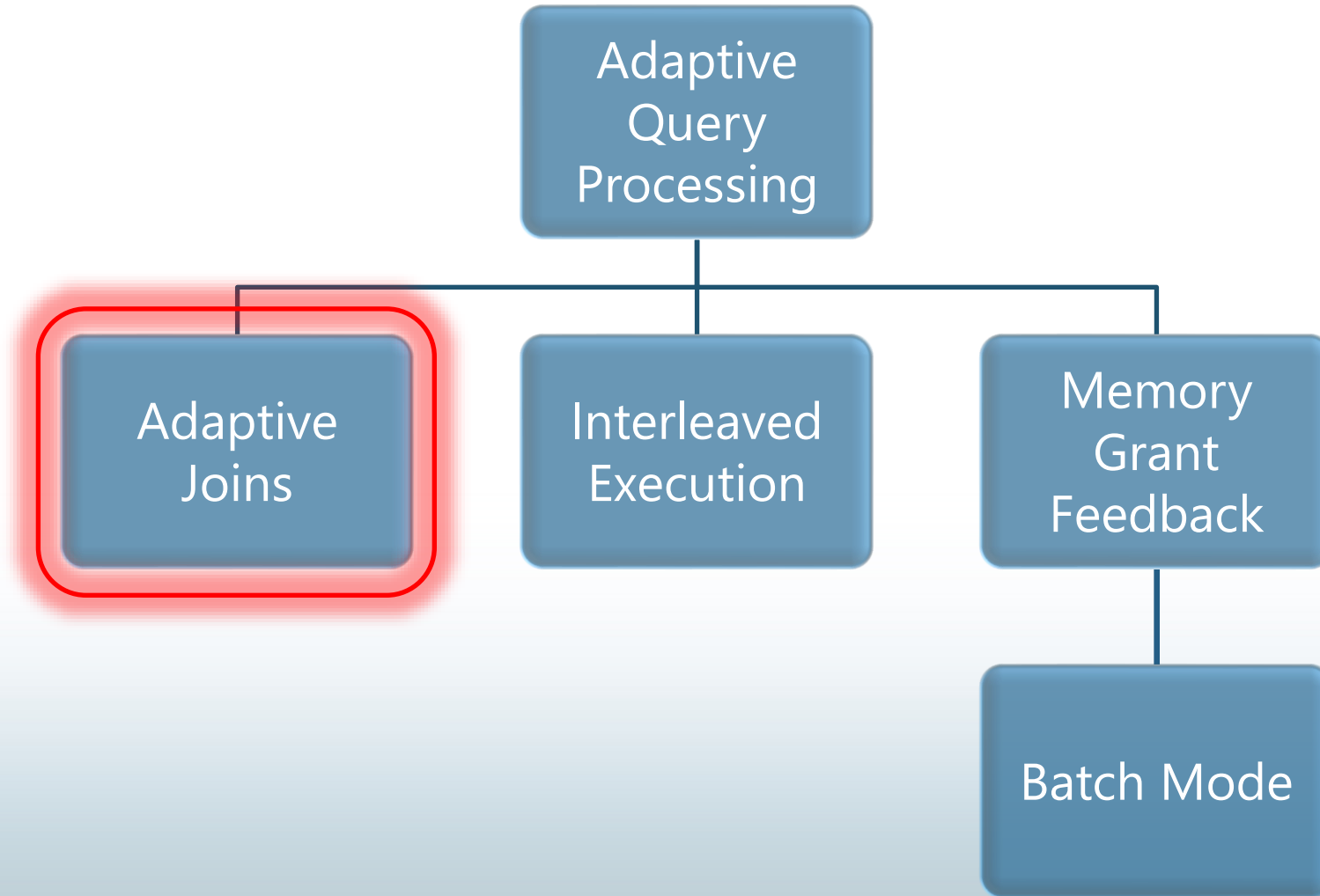
# Adaptive Query Processing (2017)



Addresses performance issues related to the cardinality estimation of an execution plan.

These options can provide improved join type selection, row-calculations for Multi-Statement Table-Valued Functions, and memory allocation of row storage.

# Batch Mode Adaptive Joins (2017)



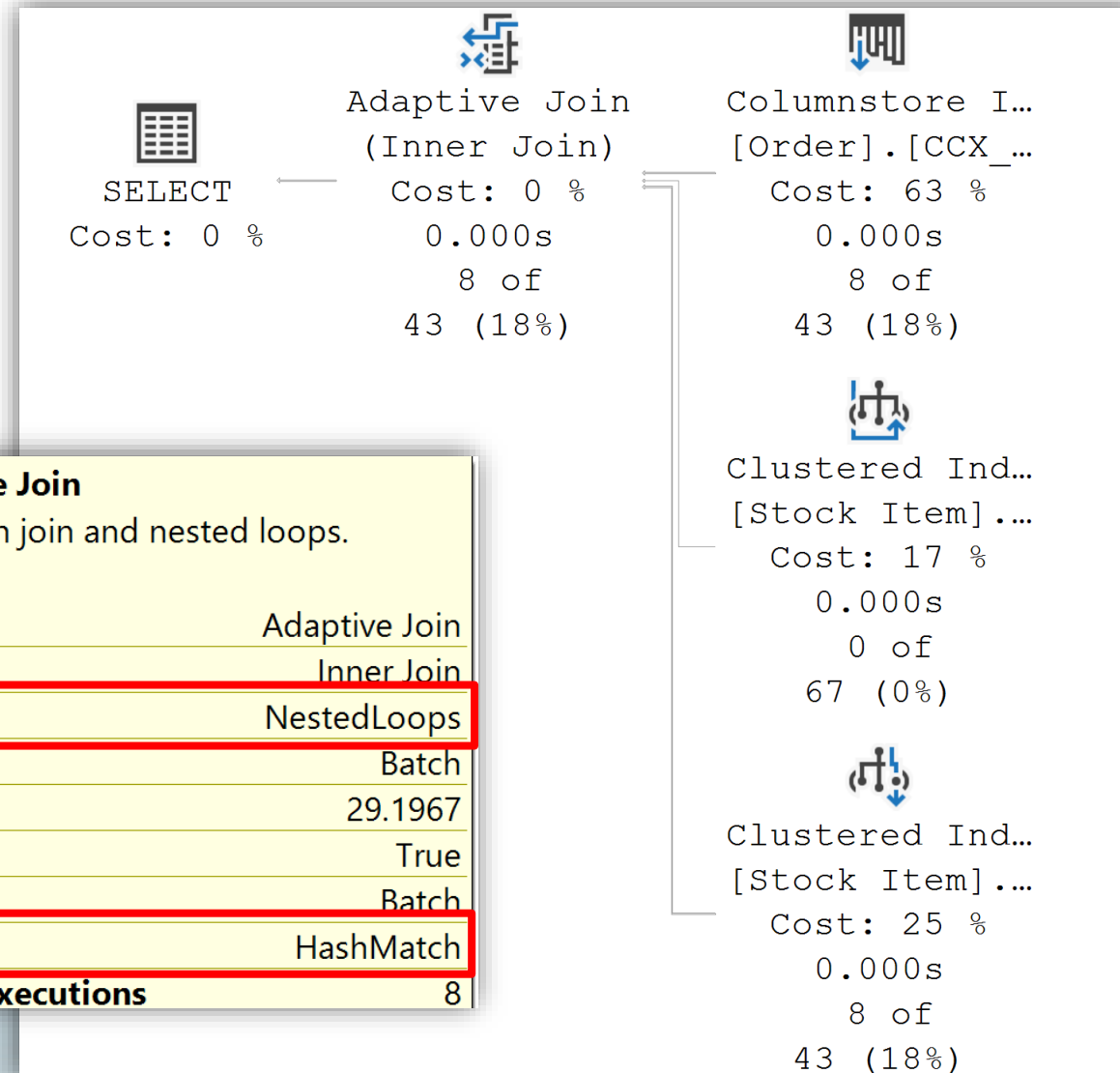
This feature enables the choice of either the Hash or the Nested Loop join type.

Decision is deferred until statement execution.

No need to use join hints in queries.

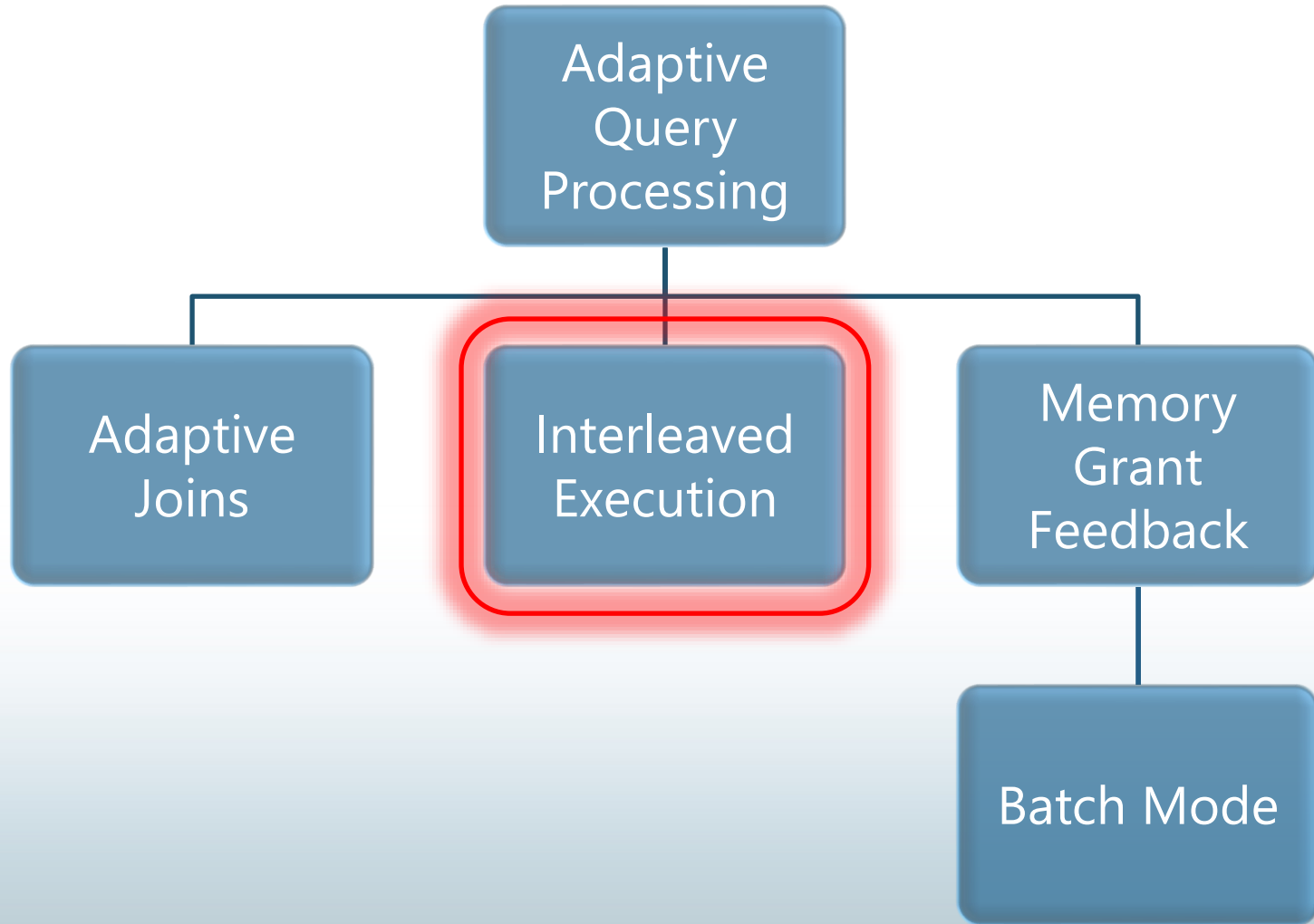
# Batch Mode Adaptive Joins (2017)

## Adaptive Joins



Adaptive Join	
Chooses dynamically between hash join and nested loops.	
Physical Operation	Adaptive Join
Logical Operation	Inner Join
Actual Join Type	NestedLoops
Actual Execution Mode	Batch
Adaptive Threshold Rows	29.1967
Is Adaptive	True
Estimated Execution Mode	Batch
Estimated Join Type	HashMatch
Actual Number of Rows for All Executions	8

# Interleaved Execution (2017)



Previously, when a Multi-Statement Table-Valued Function was executed, it used a fixed row estimate of 100 rows.

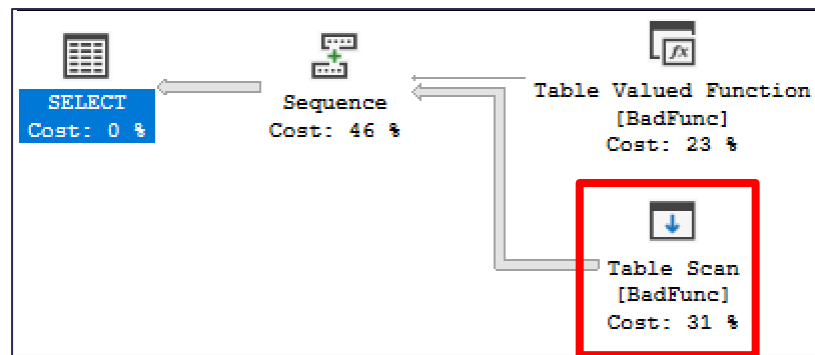
Now execution is paused so a better cardinality estimate can be captured.

# Interleaved Execution (2017)

## Interleaved Execution

### Compatibility Level 120/130

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	12345
Actual Number of Rows	12345
Actual Number of Batches	0
Estimated Operator Cost	0.003392 (92%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.000267
Estimated Subtree Cost	0.003392
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows to be Read	100
Estimated Number of Rows	100
Estimated Row Size	67 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	2



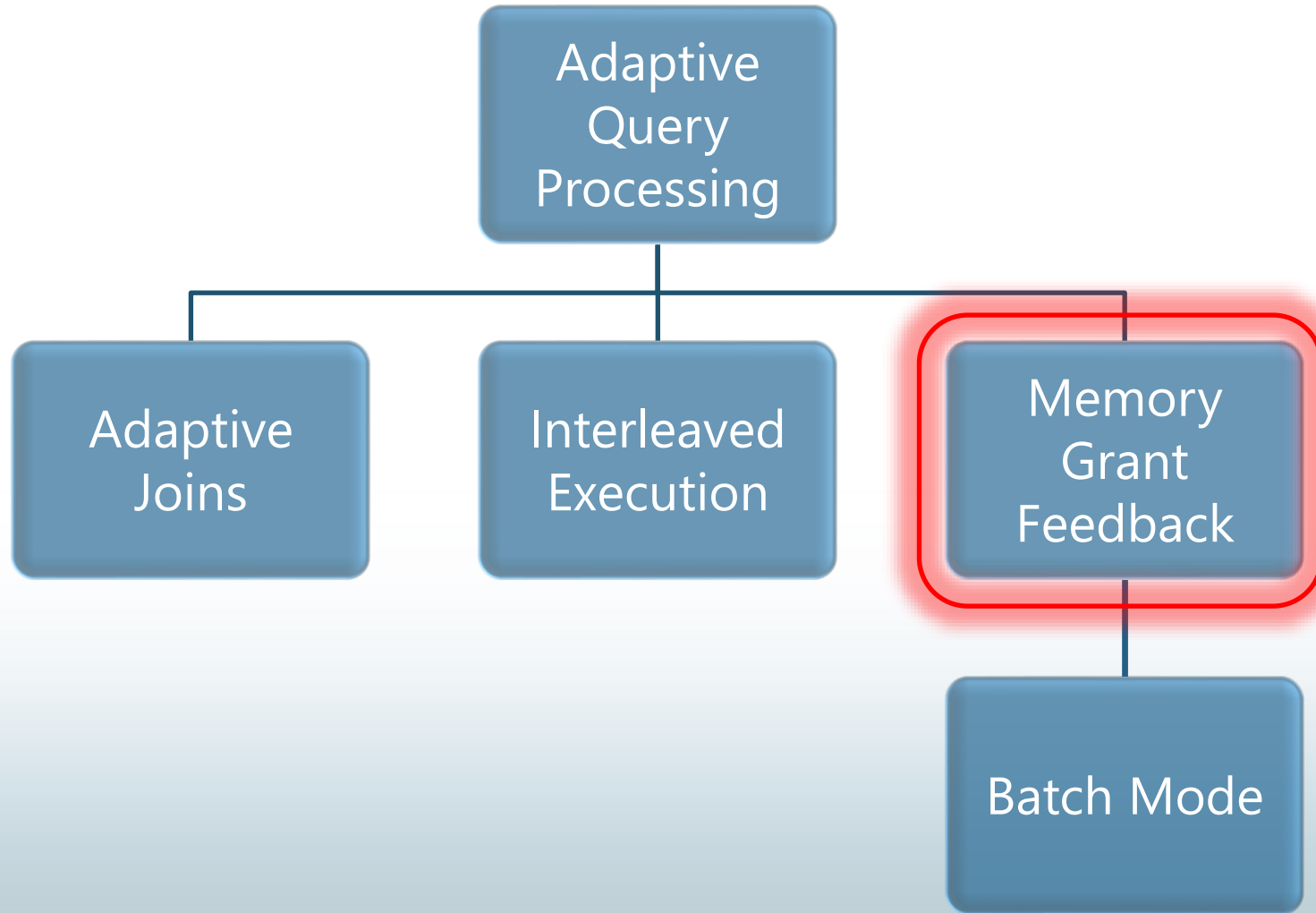
During optimization if SQL Server encounter a read-only multi-statement table-valued function (MSTVF), it will pause optimization, execute the applicable subtree, capture accurate cardinality estimates, and then resume optimization for downstream operations.

### Compatibility Level 140 or higher

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	12345
Actual Number of Rows	12345
Actual Number of Batches	0
Estimated Operator Cost	0.0168615 (31%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0137365
Estimated Subtree Cost	0.0168615
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows to be Read	12345
Estimated Number of Rows	12345
Estimated Row Size	67 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	2



# Batch Mode Memory Grant Feedback (2017)



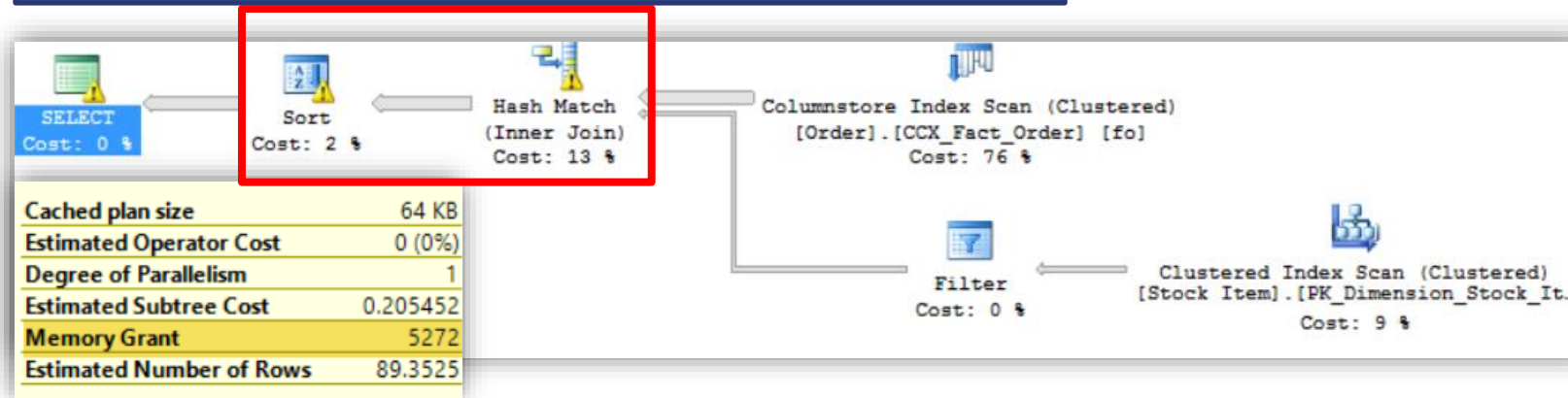
When compiling an execution plan, the query engine estimates how much memory is needed to store rows during join and sort operations.

Too much memory allocation may impact performance of other operations. Not enough will cause a spill over to disk.

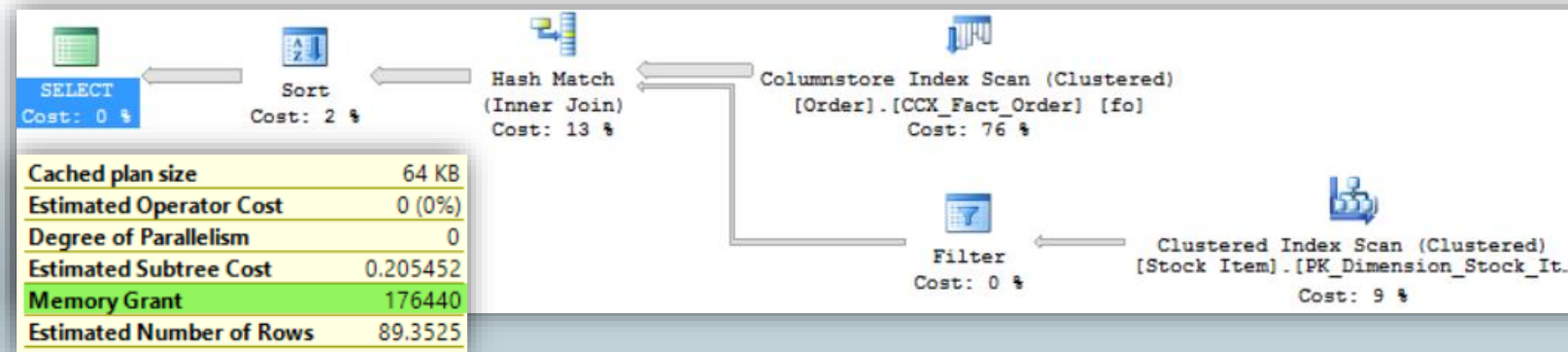
This feature recalculates memory on first execution and updates the cached plan.

# Batch Mode Memory Grant Feedback (2017)

First Execution (Spills detected; feedback generated)

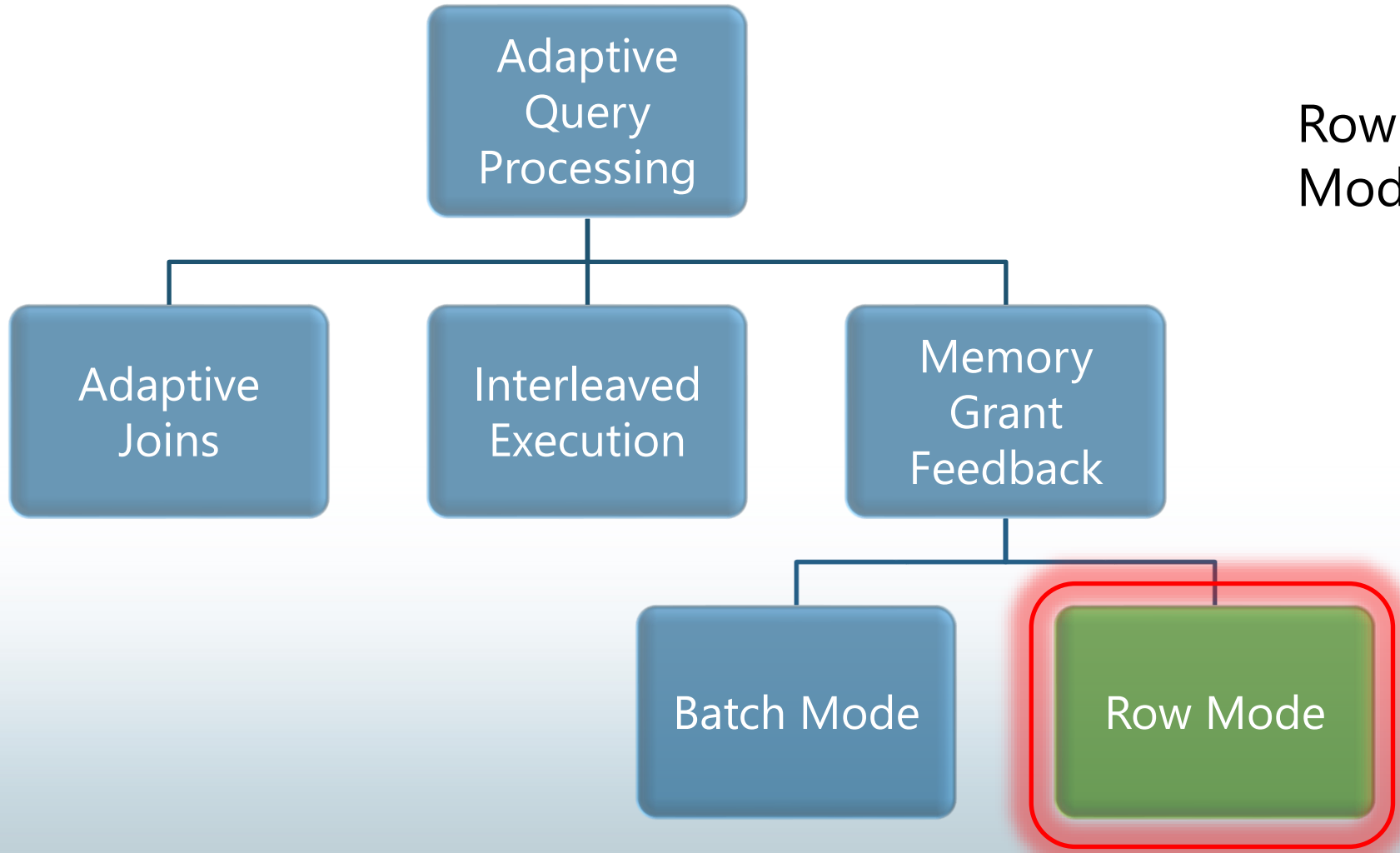


Second Execution (Memory grant adjusted)



Memory Grant  
Feedback  
(Batch Mode)

# Row Mode Memory Grant Feedback (2019)



Row Mode is just like Batch Mode, but different.

# Row Mode Memory Grant Feedback (2019)

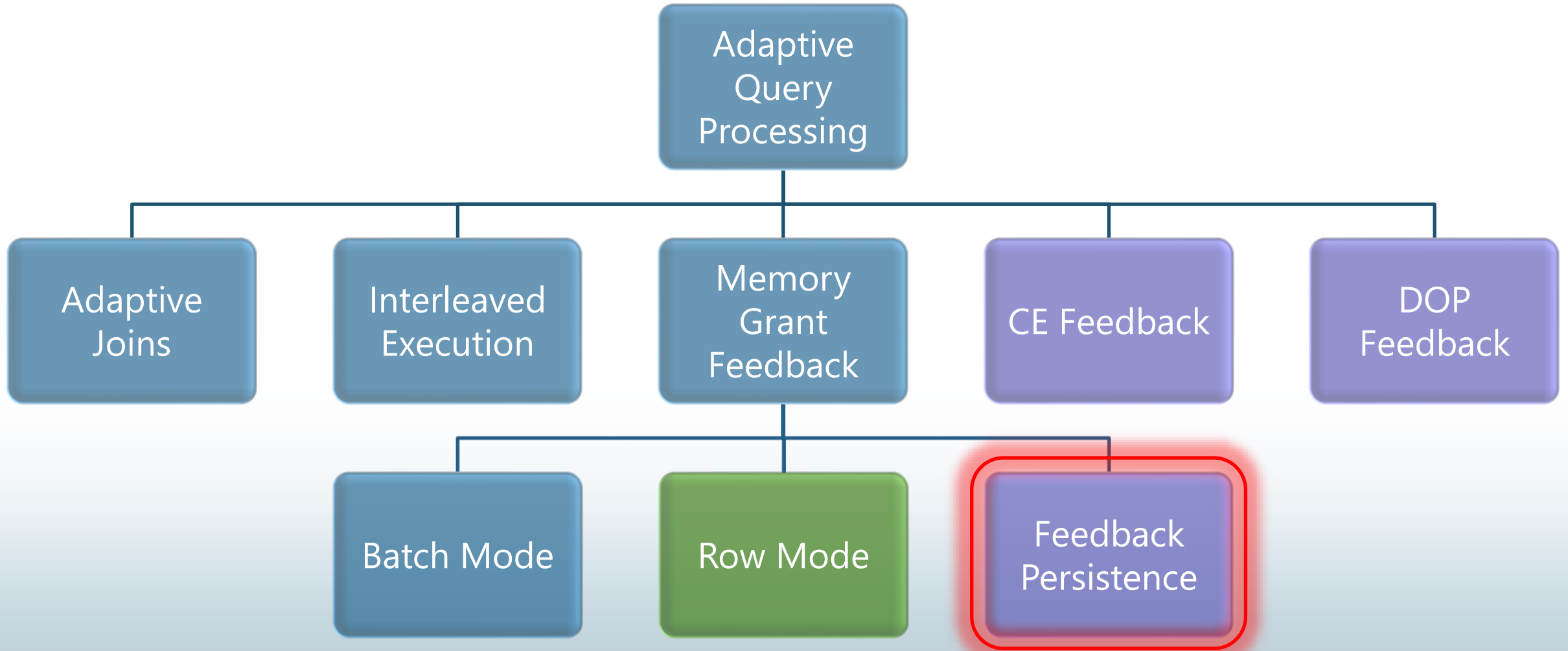
Expands on the batch mode memory grant feedback feature by also adjusting memory grant sizes for row mode operators.

MemoryGrantInfo	
DesiredMemory	13992
GrantedMemory	13992
GrantWaitTime	0
IsMemoryGrantFeedbackAdjusted	YesStable
LastRequestedMemory	13992
MaxQueryMemory	1497128
MaxUsedMemory	3744

Memory Grant  
Feedback  
(Row Mode)

Two new query plan attributes will be shown for actual post-execution plans.

# Feedback Persistence (2022)



# Feedback Persistence and Percentile (2022)

## **Problem:** Cache Eviction

- Feedback is not persisted if the plan is evicted from cache or failover
- Record of how to adjust memory is lost and must re-learn

## **Solution:** Persist the feedback

- Persist the memory grant feedback in the Query Store

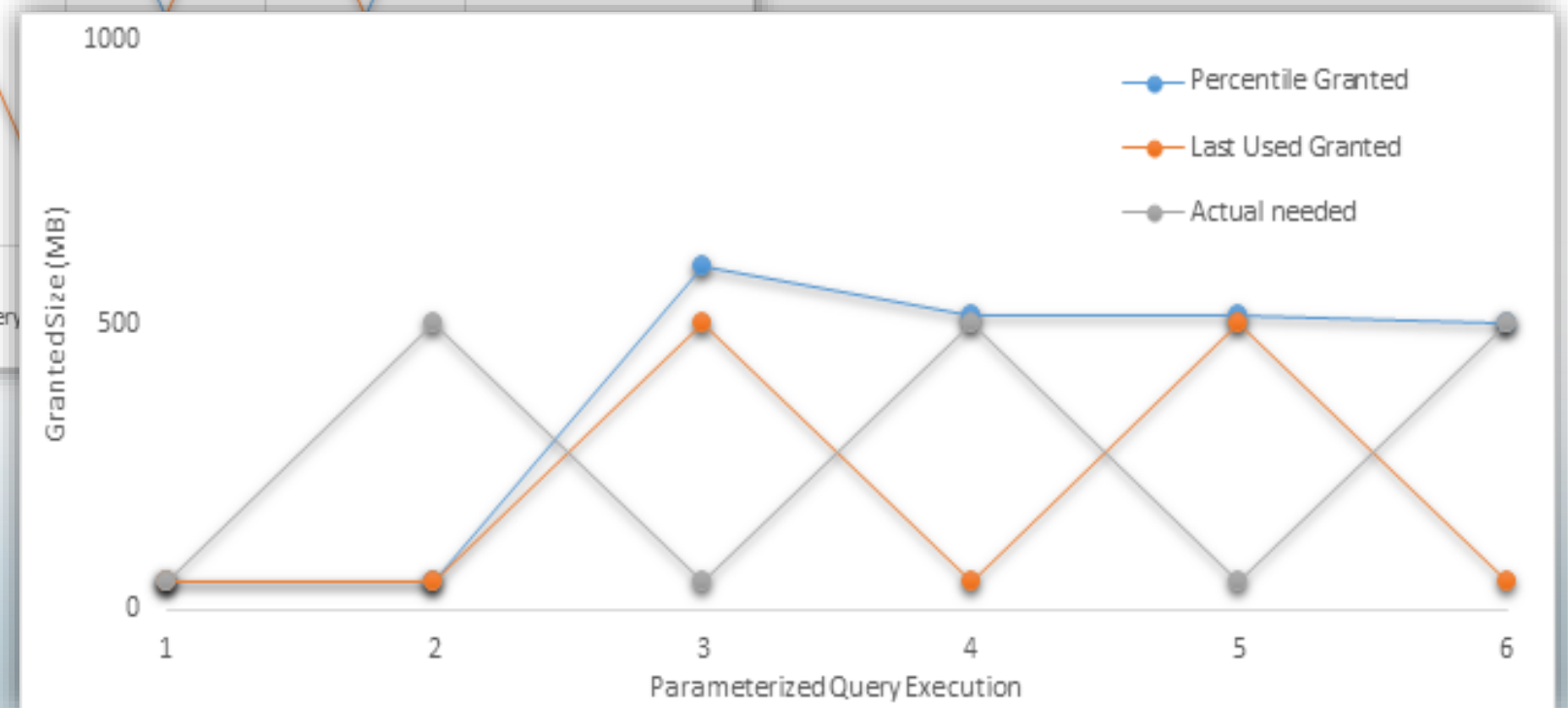
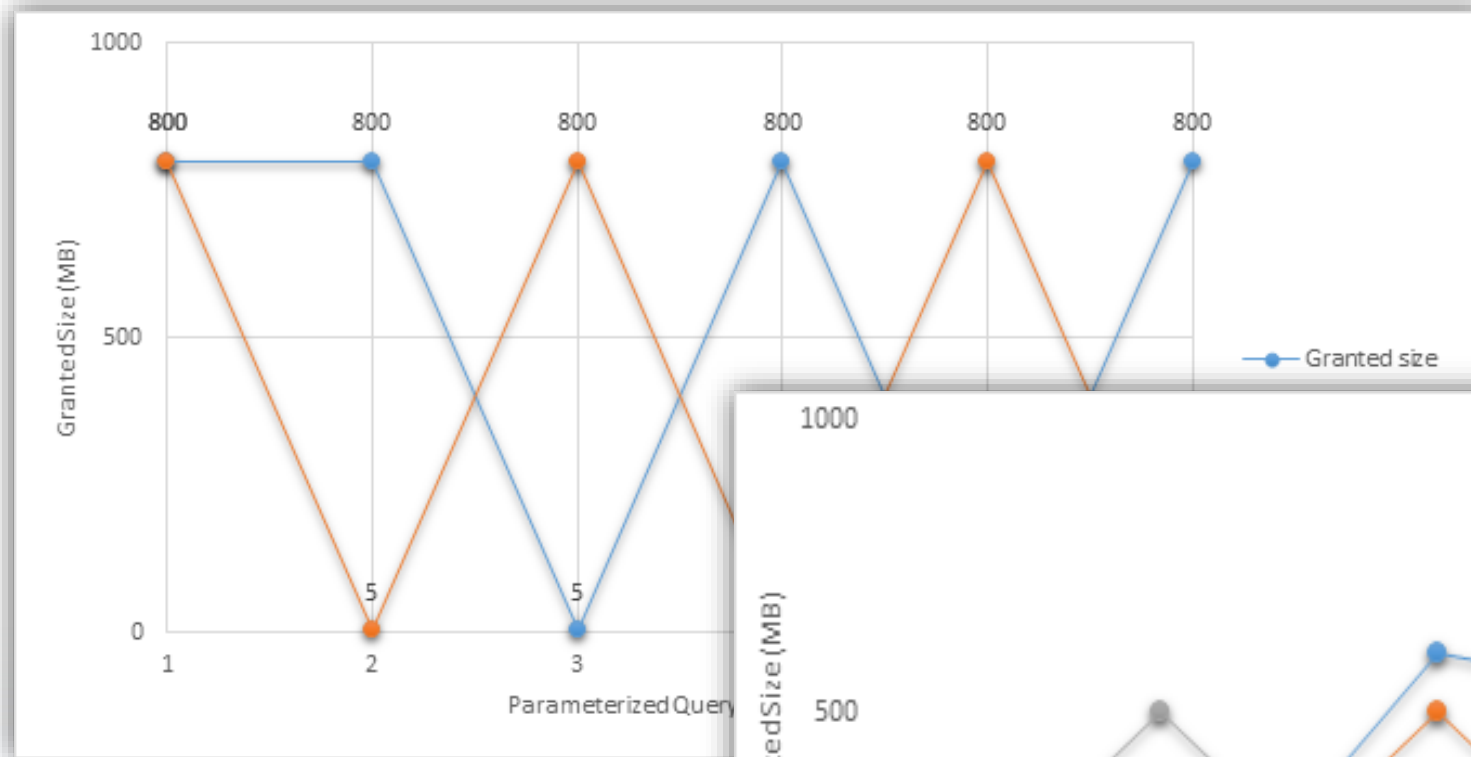
## **Problem:** Oscillating Feedback

- Memory grants adjusted based on last feedback
- Parameter Sensitive Plans could change feedback

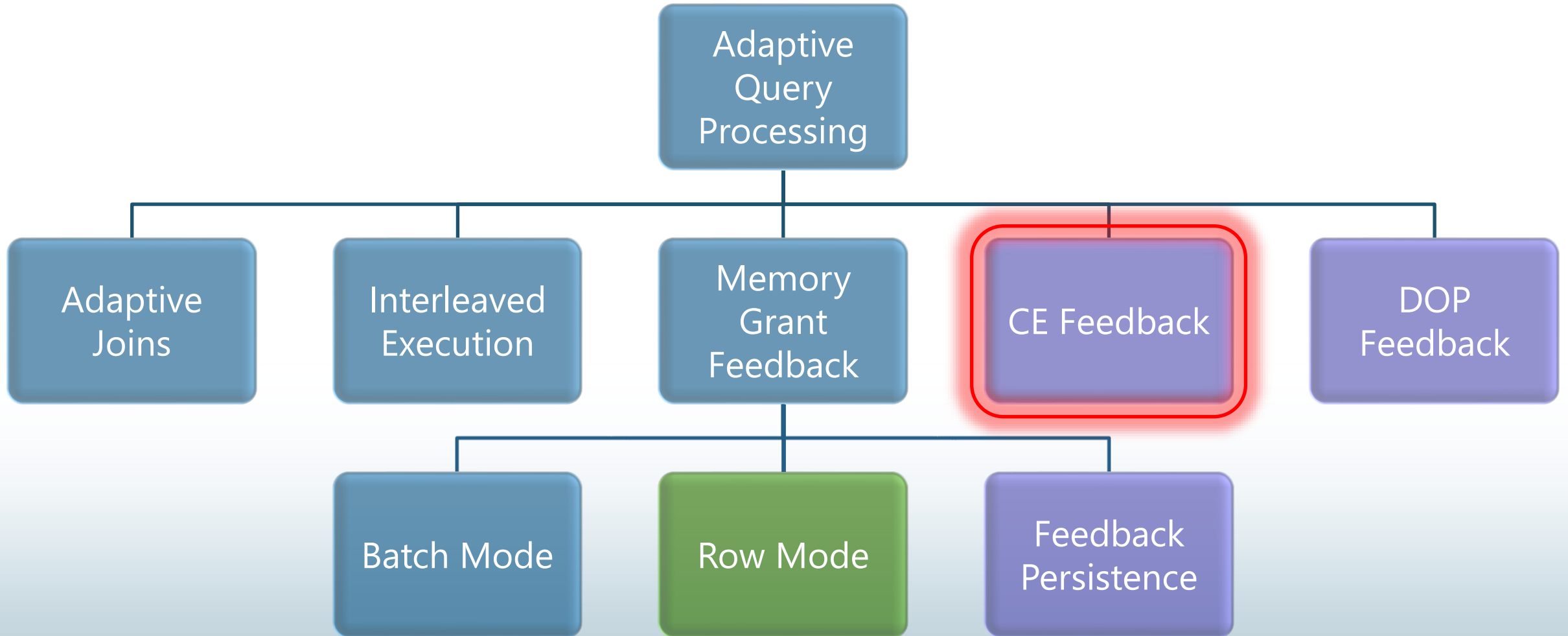
## **Solution:** Percentile-based calculation

- Smooths the grant size values based on execution usage history

# Feedback Persistence and Percentile (2022)



# Cardinality Estimator Feedback (2022)





# Cardinality Estimator Feedback (2022)

## Cardinality Estimation Today

- CE determines the estimated number of rows for a query plan
- CE models are based on statistics and assumptions about the distribution of data
- Learn more about CE models and assumptions <https://aka.ms/sqlCE>

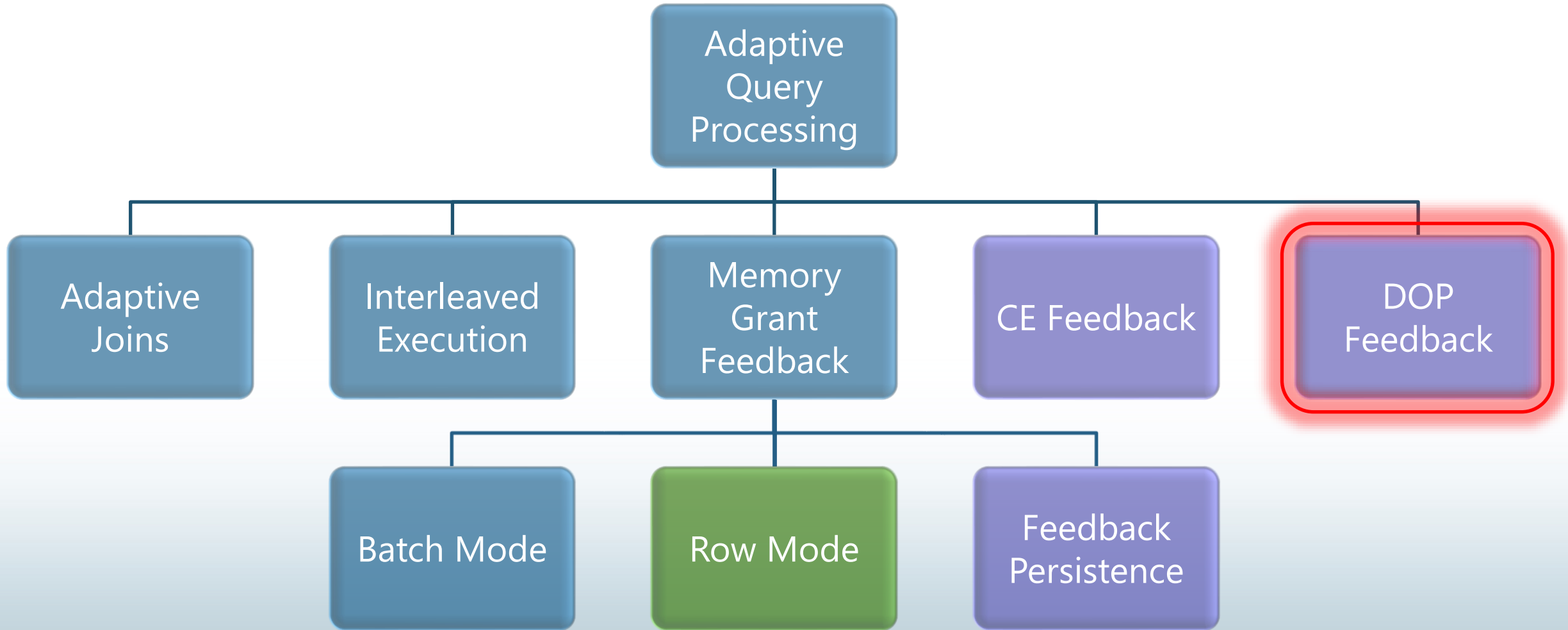
## **Problem:** Incorrect Assumptions for Cardinality Estimates

- The cardinality estimator sometimes makes incorrect assumptions
- Poor assumptions leads to poor query plans.
- One CE models doesn't fit all scenarios

## **Solution:** Learn from historical CE model assumptions

- CE Feedback will evaluate accuracy for repeated queries
- If assumption looks incorrect, test a different CE model assumption and verify if it helps
- If a CE model assumption does help, it will replace the current plan in cache.

# Degree of Parallelism Feedback (2022)



# Degree of Parallelism Feedback (2022)

## Parallelism Today

- Parallelism is often beneficial for querying large amounts of data, but transactional queries could suffer when time spent coordinating threads outweighs the advantages of using a parallel plan

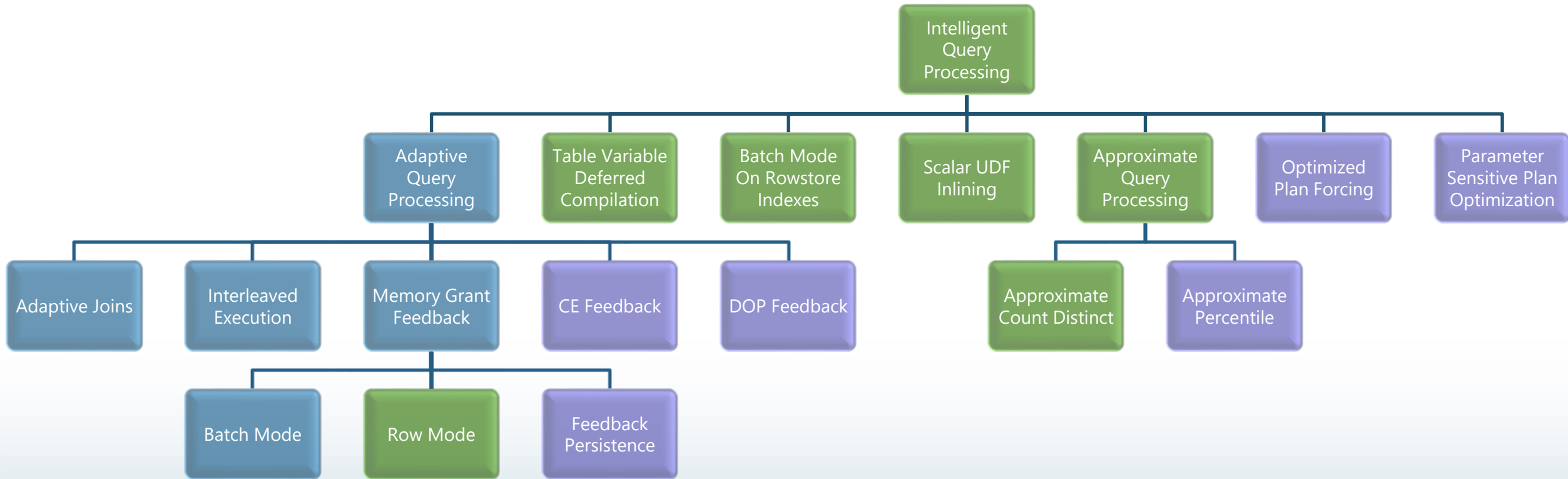
## Current Settings

- Before SQL Server 2019, default value for MAXDOP = 0
- With SQL Server 2019, default is calculated at setup based on available processors
- Azure SQL Database the default MAXDOP is 8

## DOP Feedback

- DOP Feedback will **identify** parallelism inefficiencies for repeating queries, based on CPU time, elapsed time, and waits
- If parallelism usage is inefficient, the DOP will be **lowered** for next execution (min DOP = 2) and then **verify** if it helps
- Only verified feedback is persisted (Query Store).
  - If next execution regresses, back to last good known DOP

# Intelligent Query Processing Feature Family



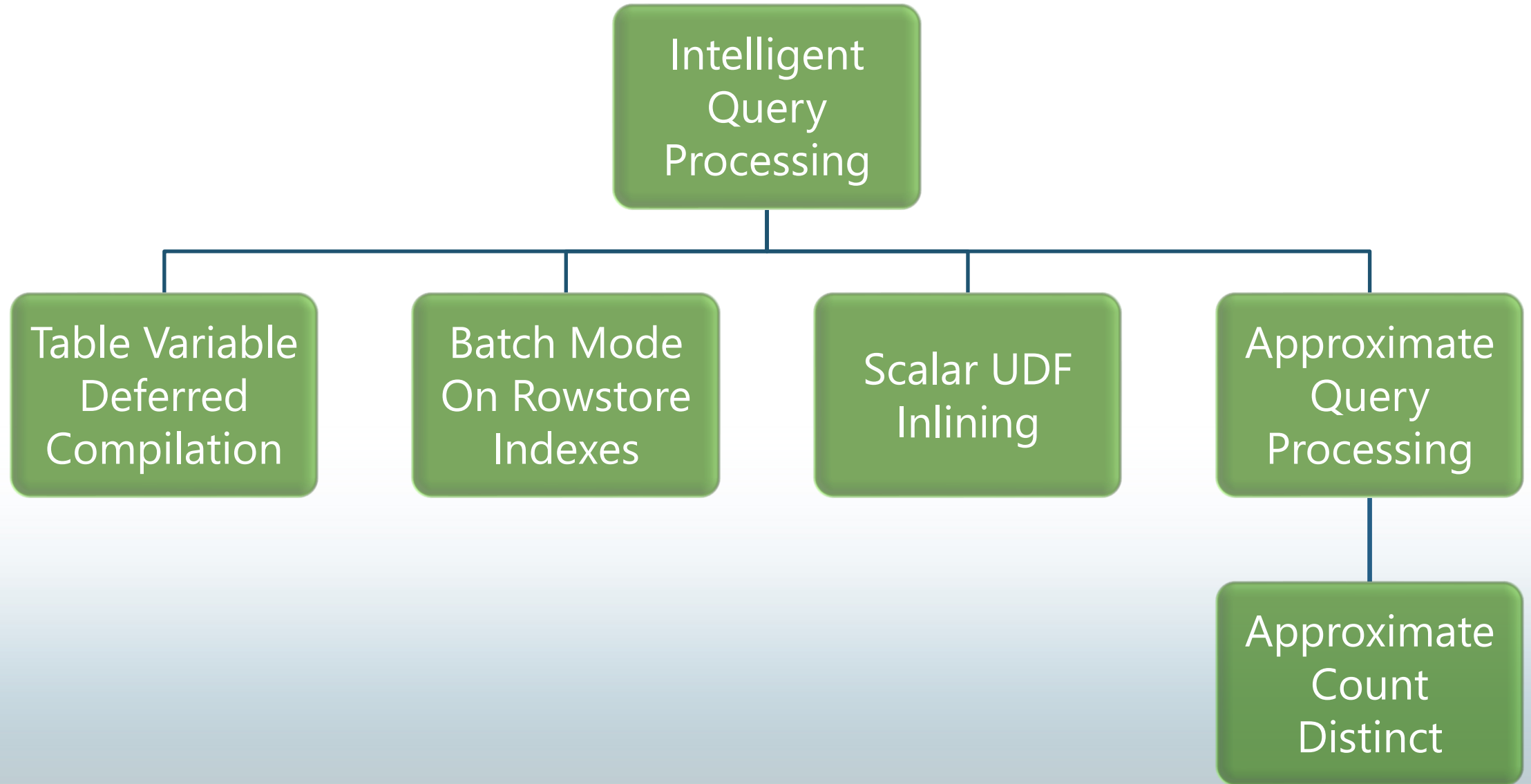
Azure SQL Database

2017

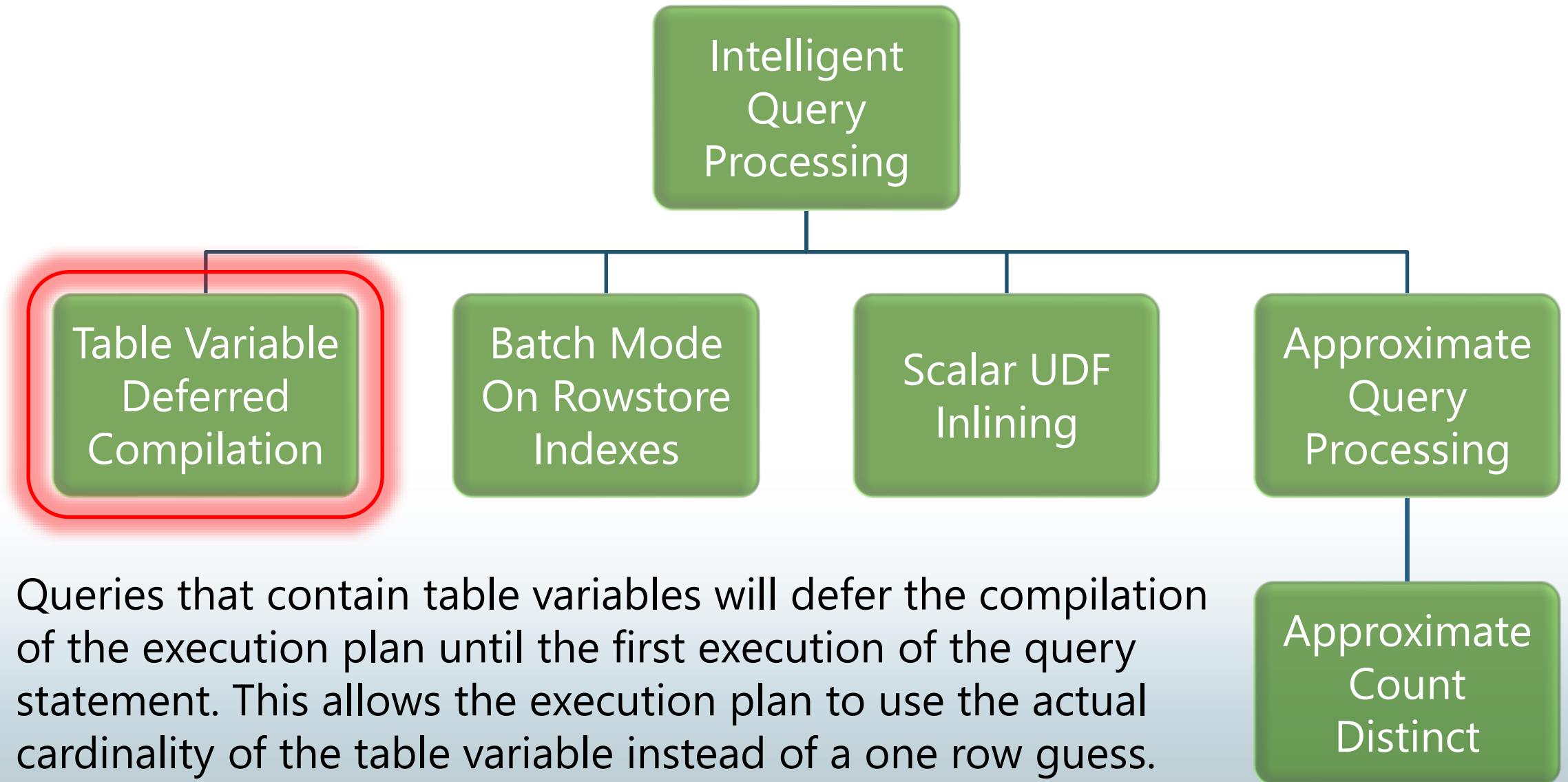
2019

2022

# Intelligent Query Processing (2019)

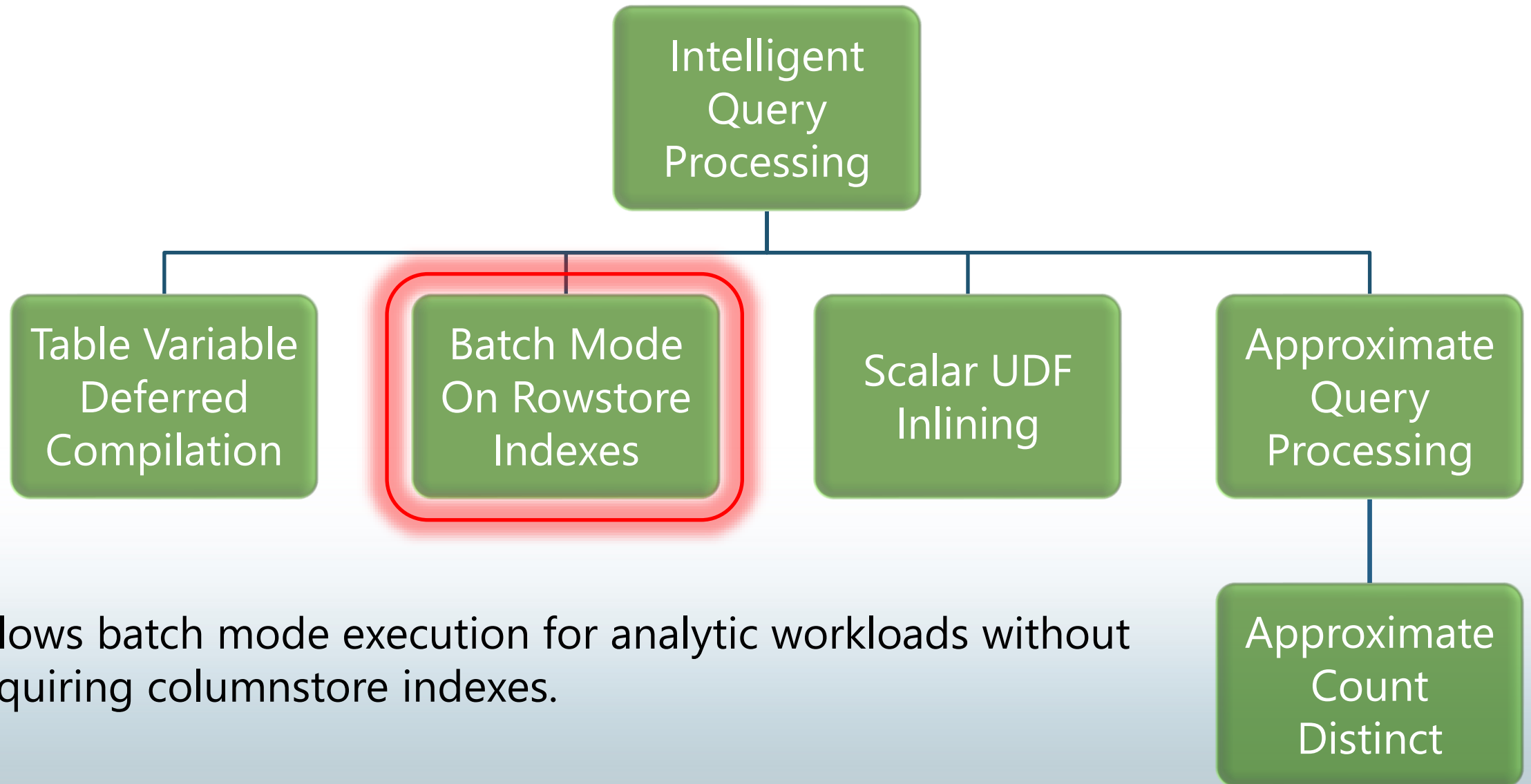


# Table Variable Deferred Compilation



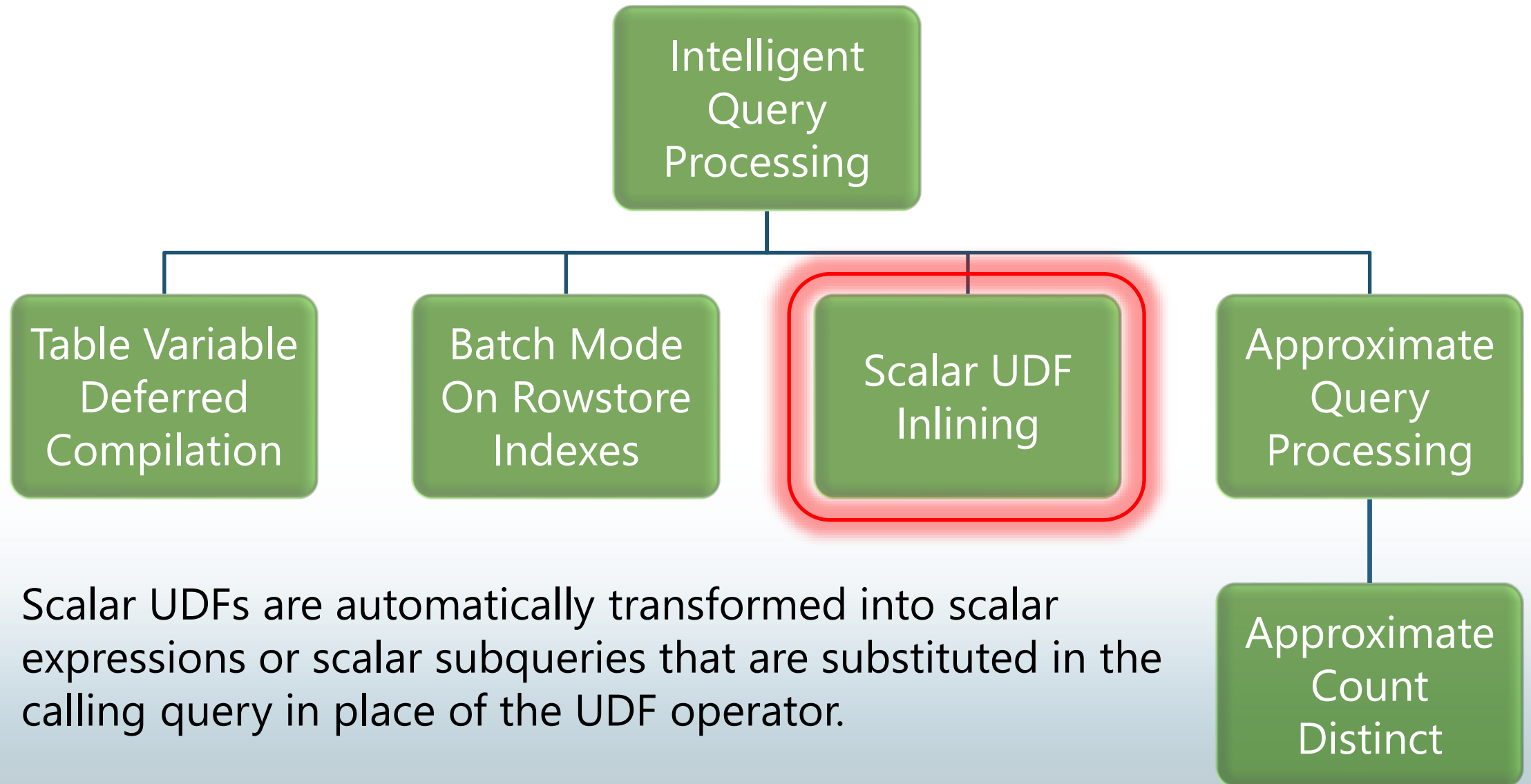
Queries that contain table variables will defer the compilation of the execution plan until the first execution of the query statement. This allows the execution plan to use the actual cardinality of the table variable instead of a one row guess.

# Batch Mode on Rowstore Indexes



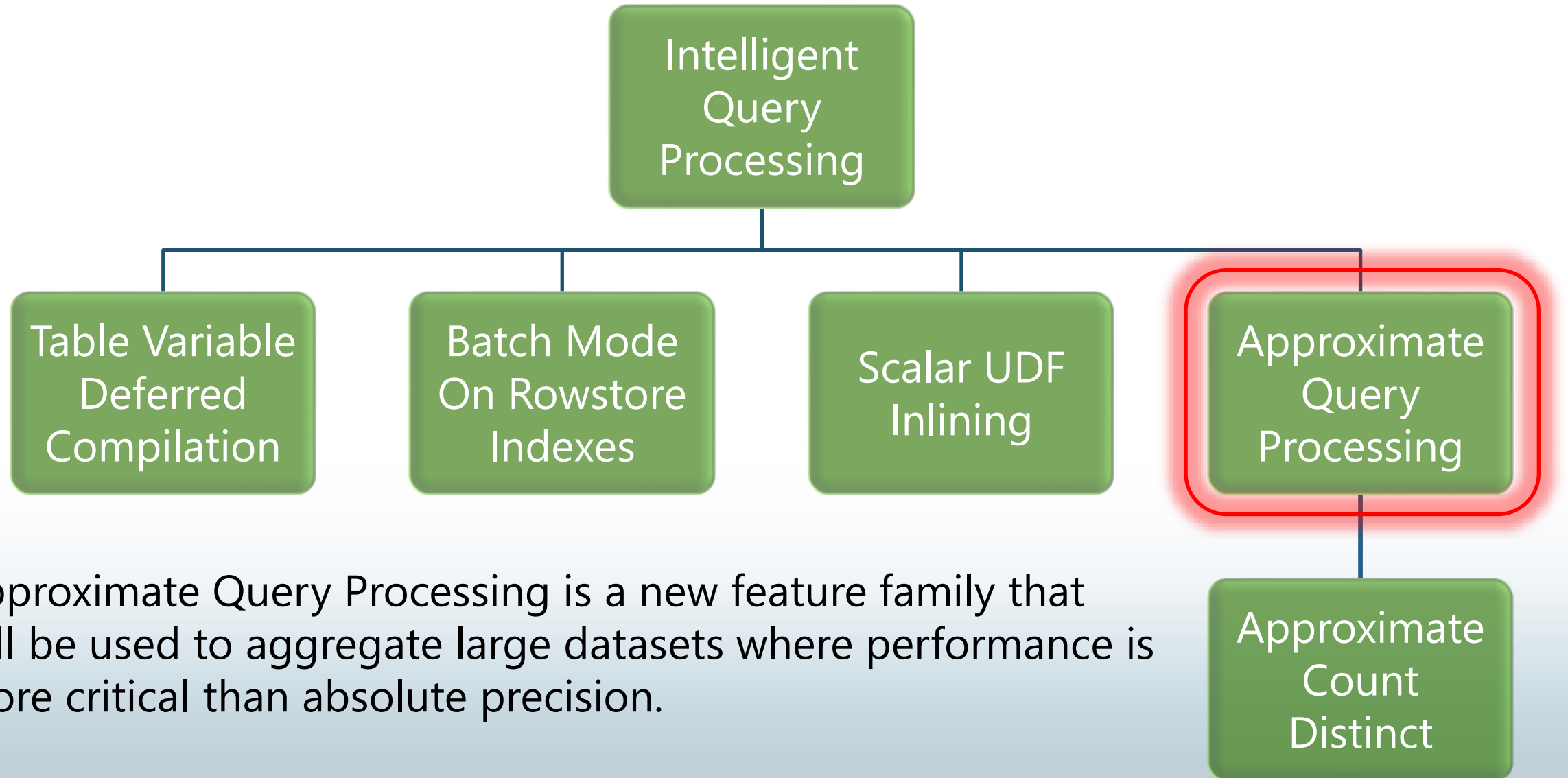
Allows batch mode execution for analytic workloads without requiring columnstore indexes.

# Scalar User-Defined Function Inlining



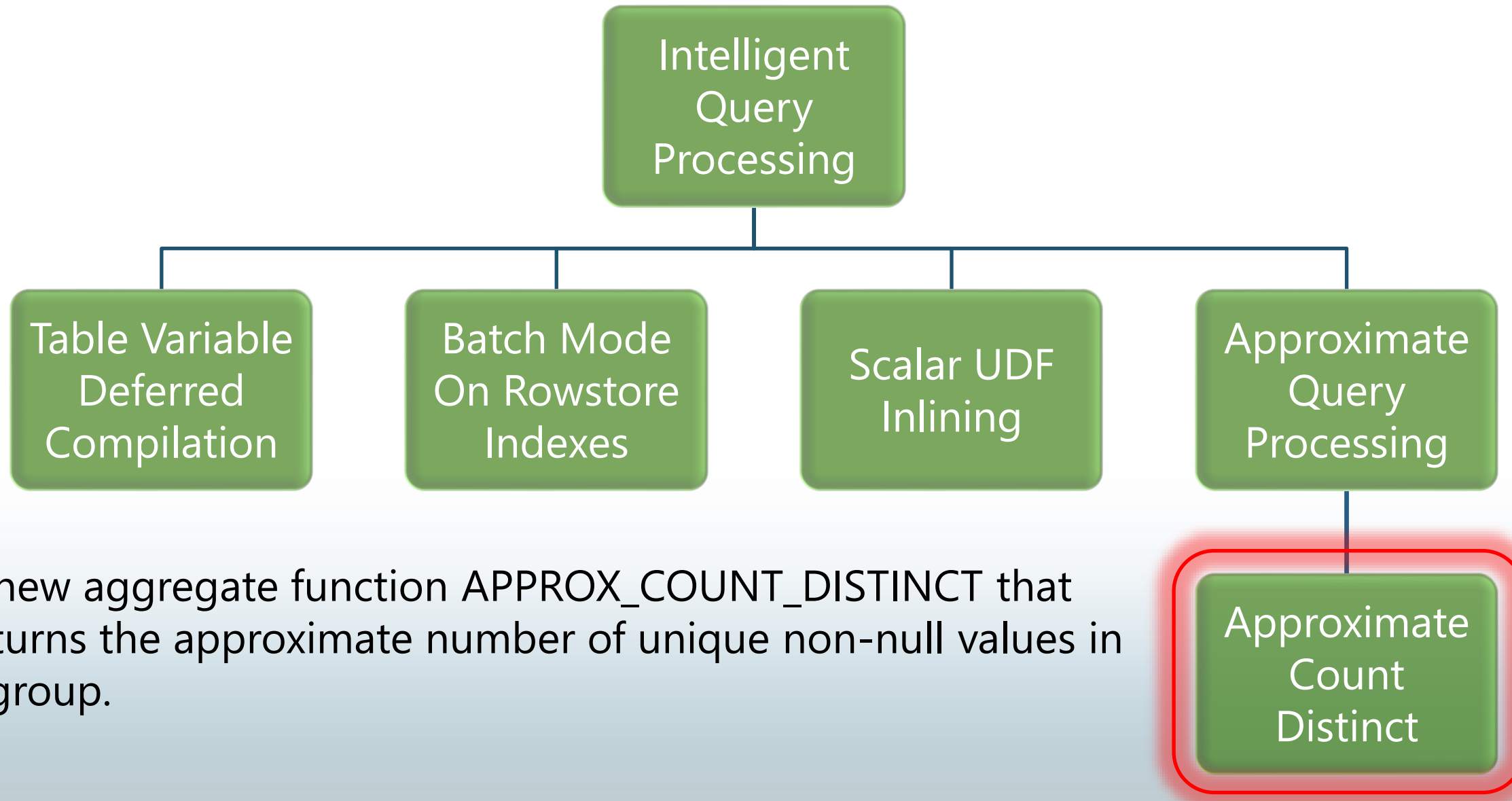


# Approximate Query Processing



Approximate Query Processing is a new feature family that will be used to aggregate large datasets where performance is more critical than absolute precision.

# Approximate Count Distinct (2019)



A new aggregate function `APPROX_COUNT_DISTINCT` that returns the approximate number of unique non-null values in a group.

# Approximate Count Distinct

It returns the approximate number of unique non-null values in a group.

It is designed to provide aggregations across large data sets where responsiveness is more critical than absolute precision.

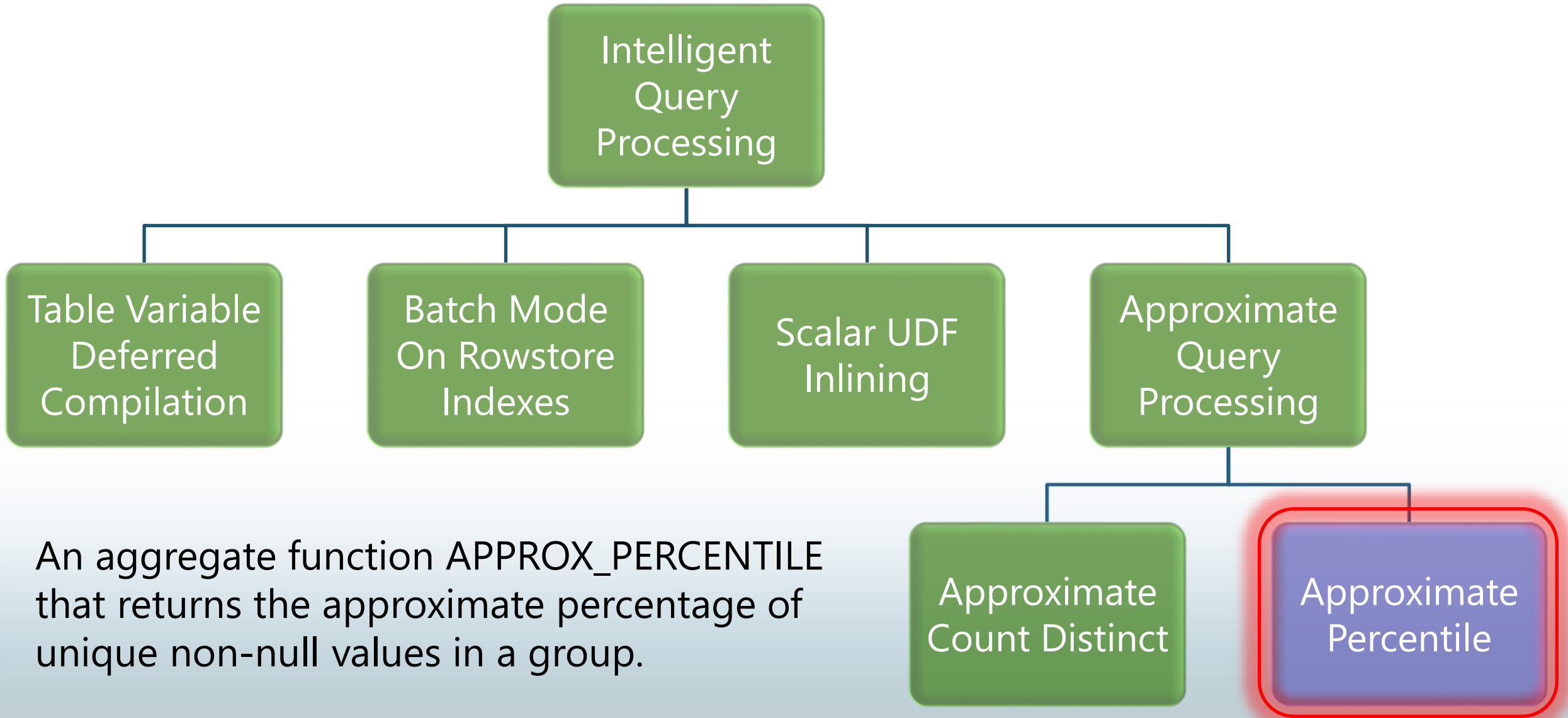
Guarantees up to a 2% error rate within a 97% probability.

Requires less memory than an exhaustive COUNT DISTINCT operation so it is less likely to spill memory to disk compared to COUNT DISTINCT.

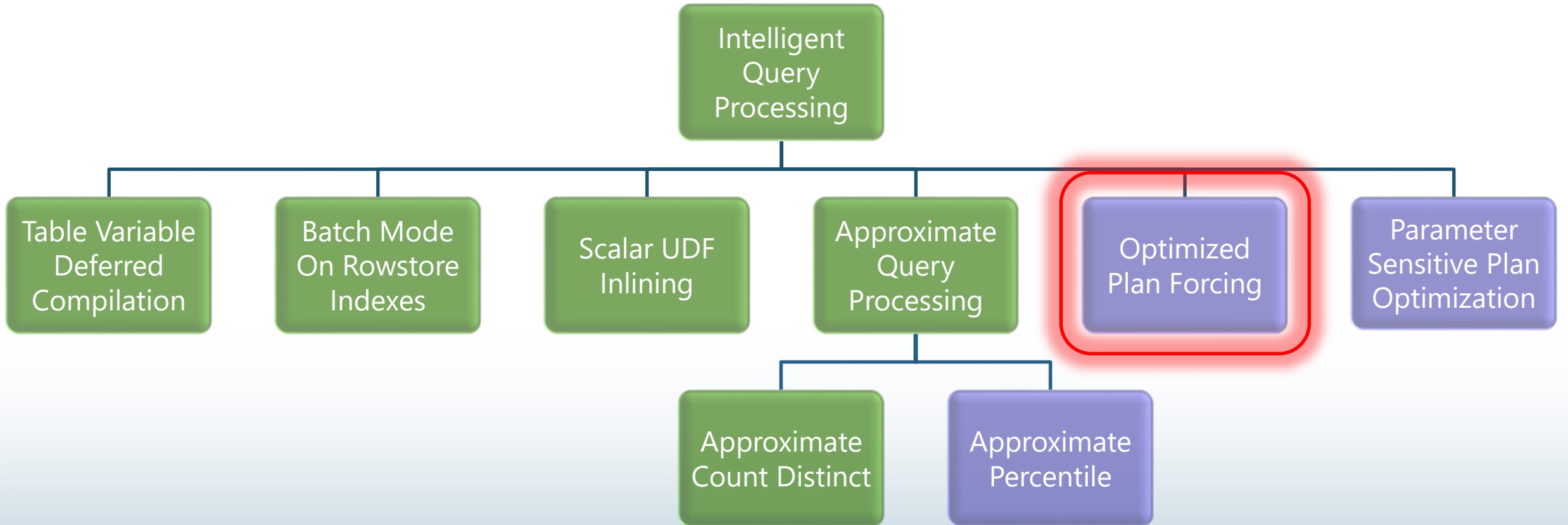
Approximate  
Count  
Distinct

```
SELECT APPROX_COUNT_DISTINCT(O_OrderKey) AS Approx_Distinct_OrderKey  
FROM dbo.Orders;
```

# Approximate Percentile (2022)



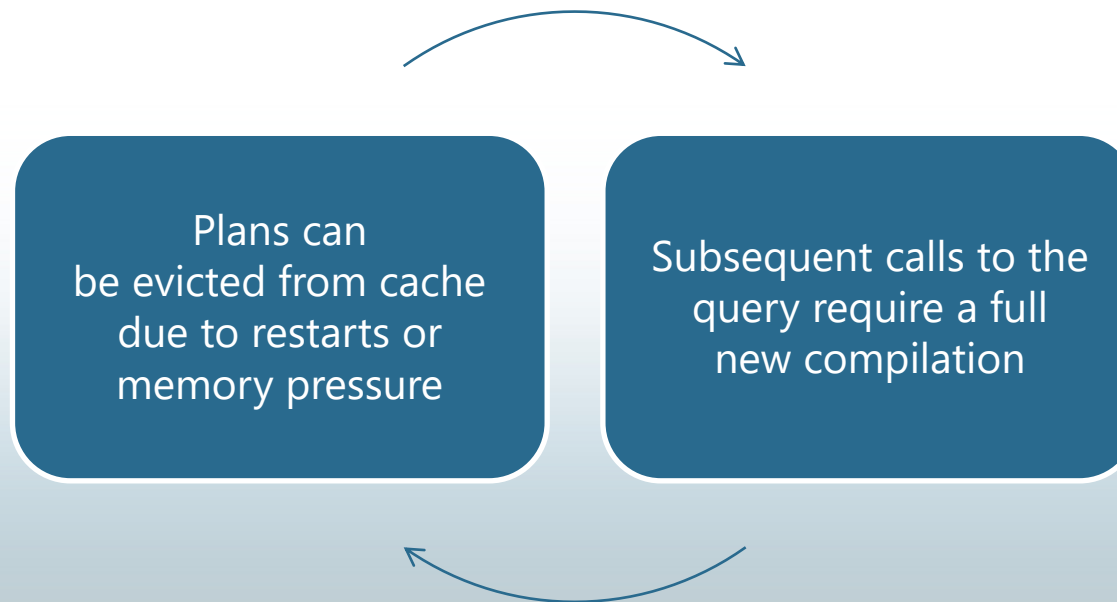
# Optimized Plan Forcing (2022)



# Optimized Plan Forcing (2022)

## Query Compilation Today

- Query optimization and compilation is a multi-phased process of quickly generating a “good-enough” query execution plan
- Query execution time includes compilation. Can be time and resource consuming
- To reduce compilation overhead for repeating queries, SQL caches query plans for re-use

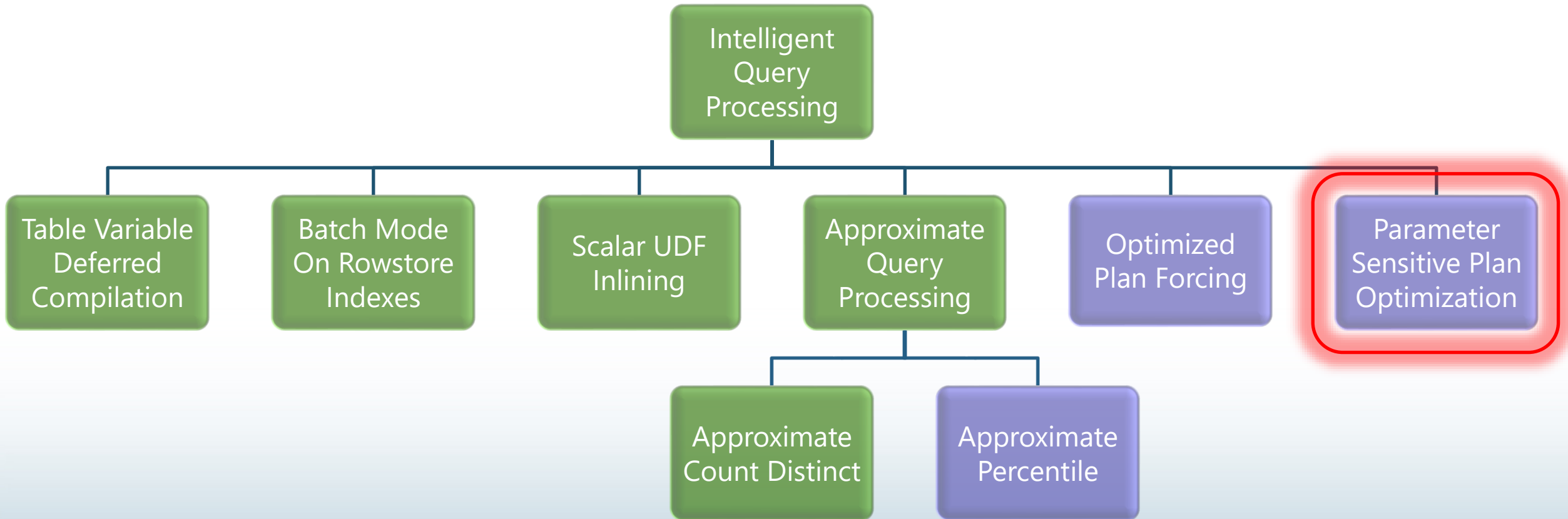


# Optimized Plan Forcing (2022)

## Query Compilation Replay

- Stores a *compilation replay script* (CRS) that persists key compilation steps in Query Store (not user visible)
- Version 1 targets previously forced plans through Query Store and Automatic Plan Correction
- Uses those previously-recorded CRS to quickly reproduce and cache the original forced plan **at a fraction of the original compilation cost**
- Compatible with Query Store hints and secondary replica support

# Parameter Sensitive Plan Optimization (2022)





# Parameter Sensitive Plans (2022)

## Parameter Sensitive Plans Today

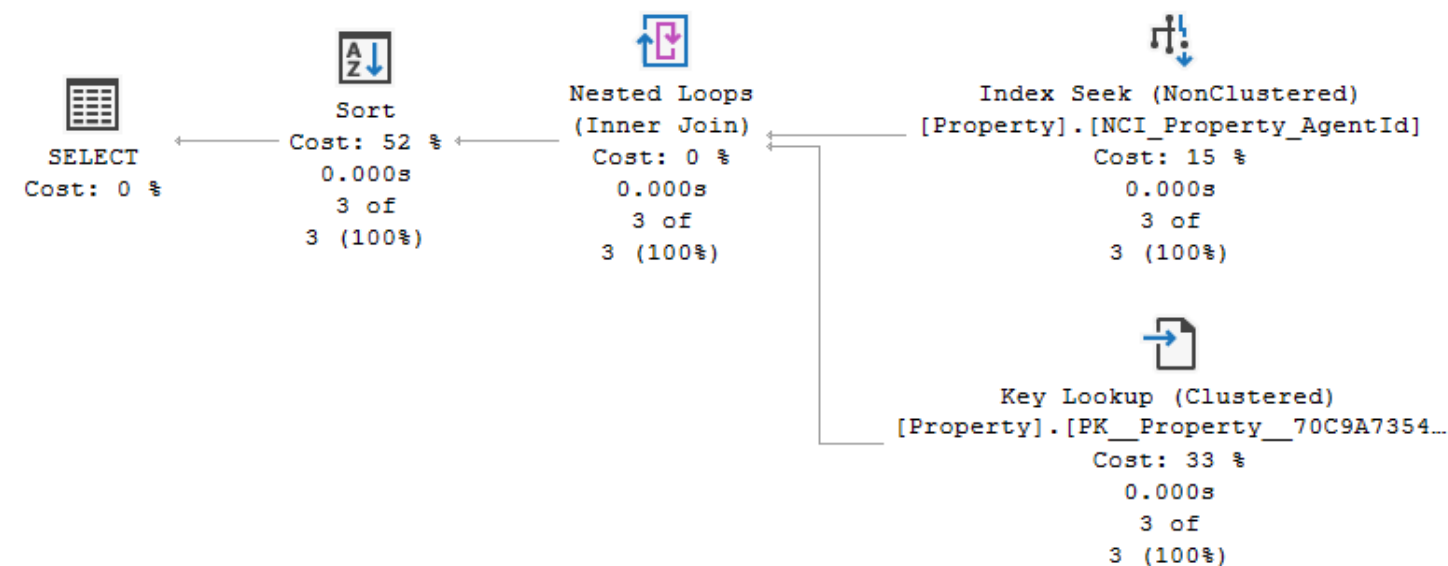
- Parameter-sniffing problem refers to a scenario where a **single** cached plan for a parameterized query is **not optimal for all** possible input parameter values
- If plan is not representative of most executions, you have a perceived “bad plan”

## Current Workarounds

- RECOMPILE
- OPTION (OPTIMIZE FOR...)
- OPTION (OPTIMIZE FOR UNKNOWN)
- Disable parameter sniffing entirely
- KEEPFIXEDPLAN
- Force a known plan
- Nested procedures
- Dynamic string execution

# PSP today (Example of Real Estate agent's portfolio)

New compile on Agent 4

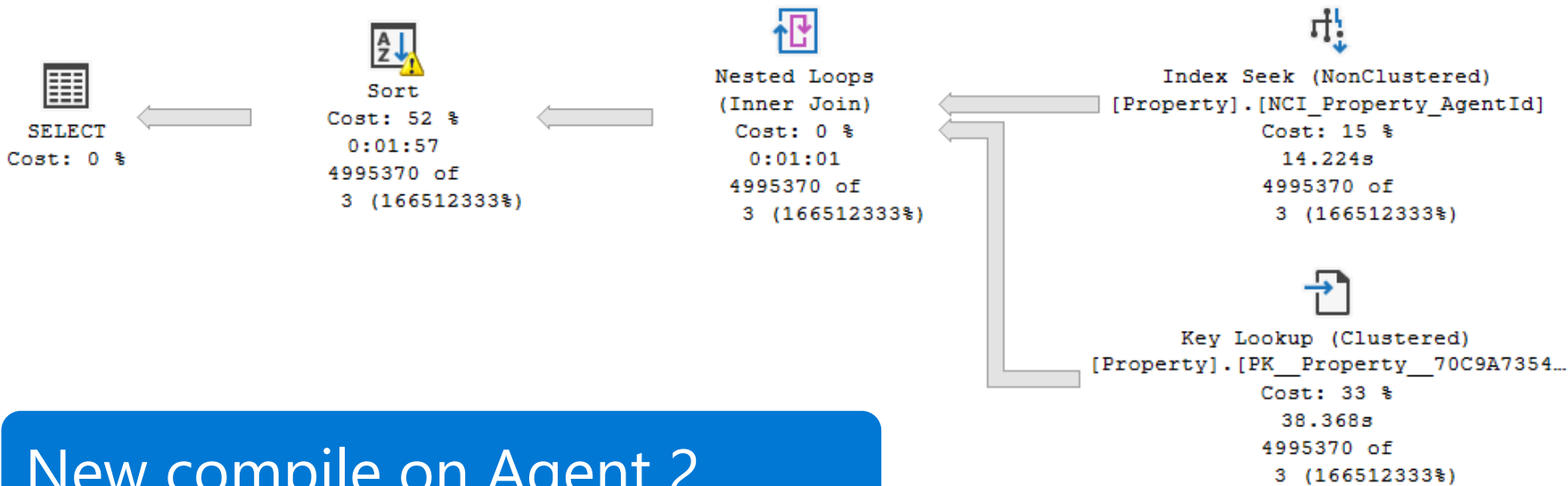


QueryTimeStats	
CpuTime	0
ElapsedTime	0

This example was borrowed from Pedro Lopes @SQLPedro

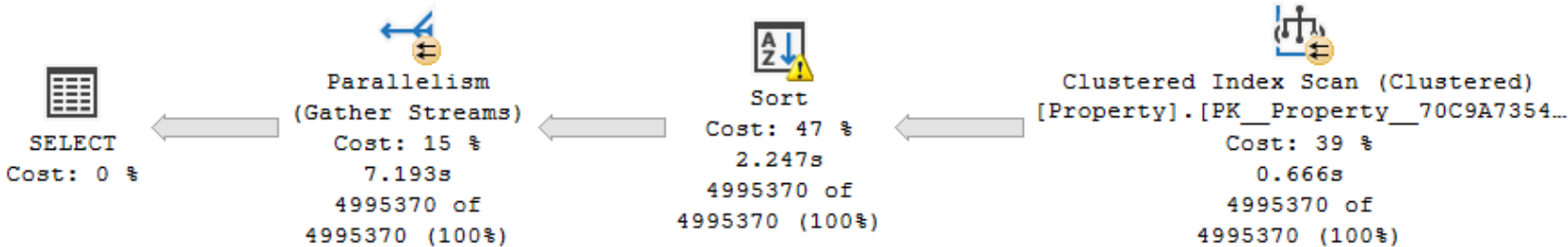
# PSP today (Example of Real Estate agent's portfolio)

Using cached plan for Agent 2



QueryTimeStats	
CpuTime	88667
ElapsedTime	214222

New compile on Agent 2



QueryTimeStats	
CpuTime	46620
ElapsedTime	105288

# PSP Optimization (2022)

Automatically enables multiple, active cached plans for a single parameterized statement

Cached execution plans will accommodate different data sizes based on the customer-provided runtime parameter value(s)

## Design considerations

- Too many plans generated could create cache bloat, so limit # of plans in cache
- Overhead of PSP optimization must not outweigh downstream benefit
- Compatible with Query Store plan forcing

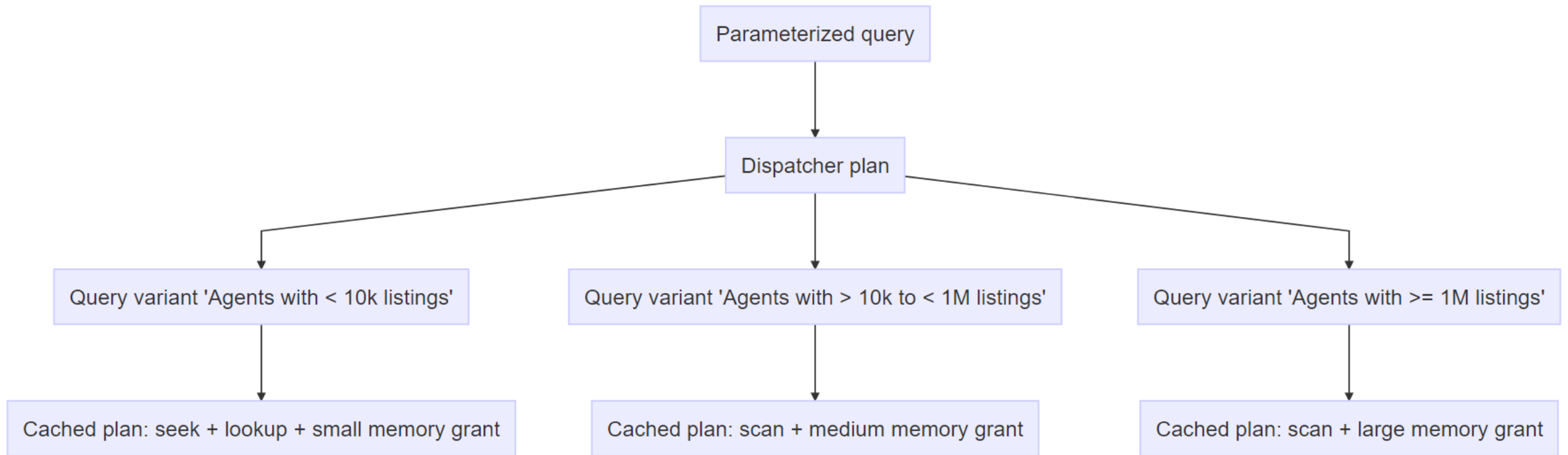
# PSP Predicate Selection (2022)

During initial compilation PSP optimization will evaluate the most “at risk” parameterized predicates (up to three out of all available)

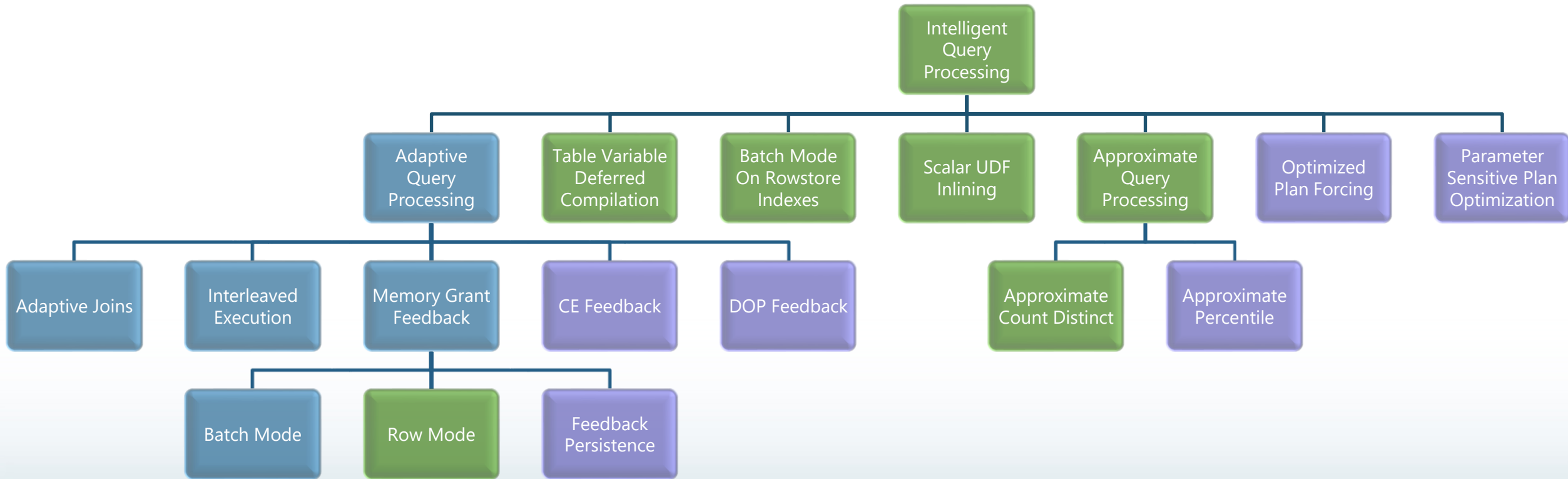
First version is scoped to equality predicates referencing statistics-covered columns; `WHERE AgentId = @AgentId`

Uses the statistics histogram to identify non-uniform distributions

# Boundary Value Selection (Dispatcher Plan)



# Intelligent Query Processing Feature Family



Azure SQL Database

2017

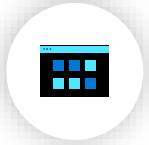
2019

2022

# Questions?



# Learn more



Learn more about SQL Server 2022

[aka.ms/sqlserver2022](https://aka.ms/sqlserver2022)



Watch a technical deep-dive on SQL Server 2022

[aka.ms/sqlserver2022mechanics](https://aka.ms/sqlserver2022mechanics)



Don't miss out on Data Exposed (Look for SQL Server 2022 – Episode 3)

[aka.ms/dataexposed](https://aka.ms/dataexposed)



More information on Intelligent Query Processing  
(Currently does not have 2022 information.)

[aka.ms/IQP](https://aka.ms/IQP)

<https://aka.ms/IQPDemos>

<https://aka.ms/SQLCE>

Dankie Faleminderit **Shukran** Chnorakaloutioun Hvala Blagodaria

Děkuji **Tak** Dank u Tānan Kiitos **Merci** Danke Ευχαριστώ A dank

Mahalo הודות **Dhanyavād** Köszönöm Takk Terima kasih **Grazie** Grazzi

**Thank you!**

감사합니다 Paldies Choukrane Ačiū **Благодарам** ありがとうございます

谢谢 Баярлалаа **Dziękuję** Obrigado Mulțumesc **Спасибо** Ngiyabonga

Ďakujem Tack Nandri Kop khun **Teşekkür ederim** Дякую Хвала Diolch