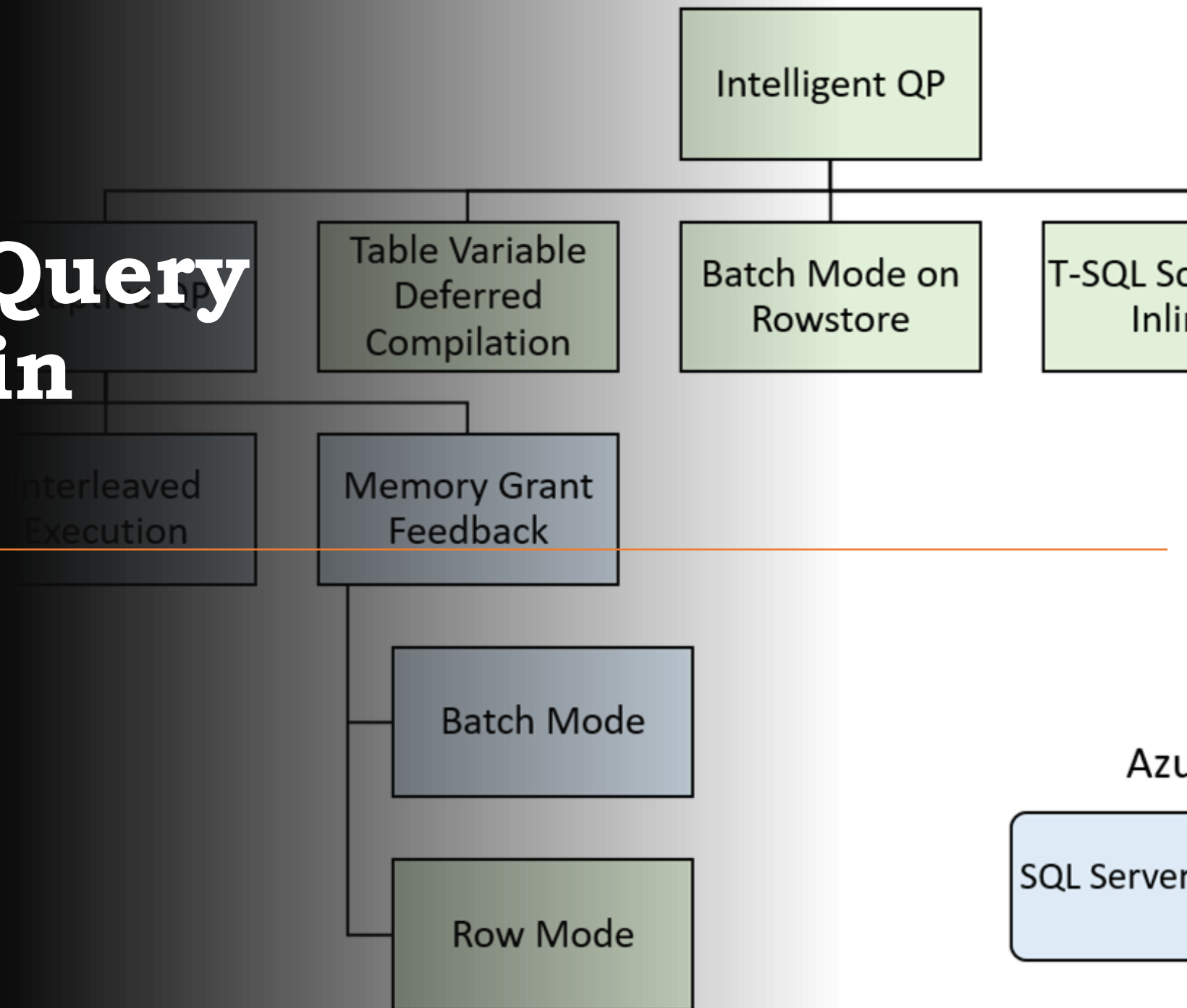


# Intelligent Query Processing in SQL Server

Taib Ali





## Data Solutions Manager, GMO LLC



<http://sqlworldwide.com/>



/sqlworldwide



@sqlworldwide



taio@sqlworldwide.com

### Data Professional

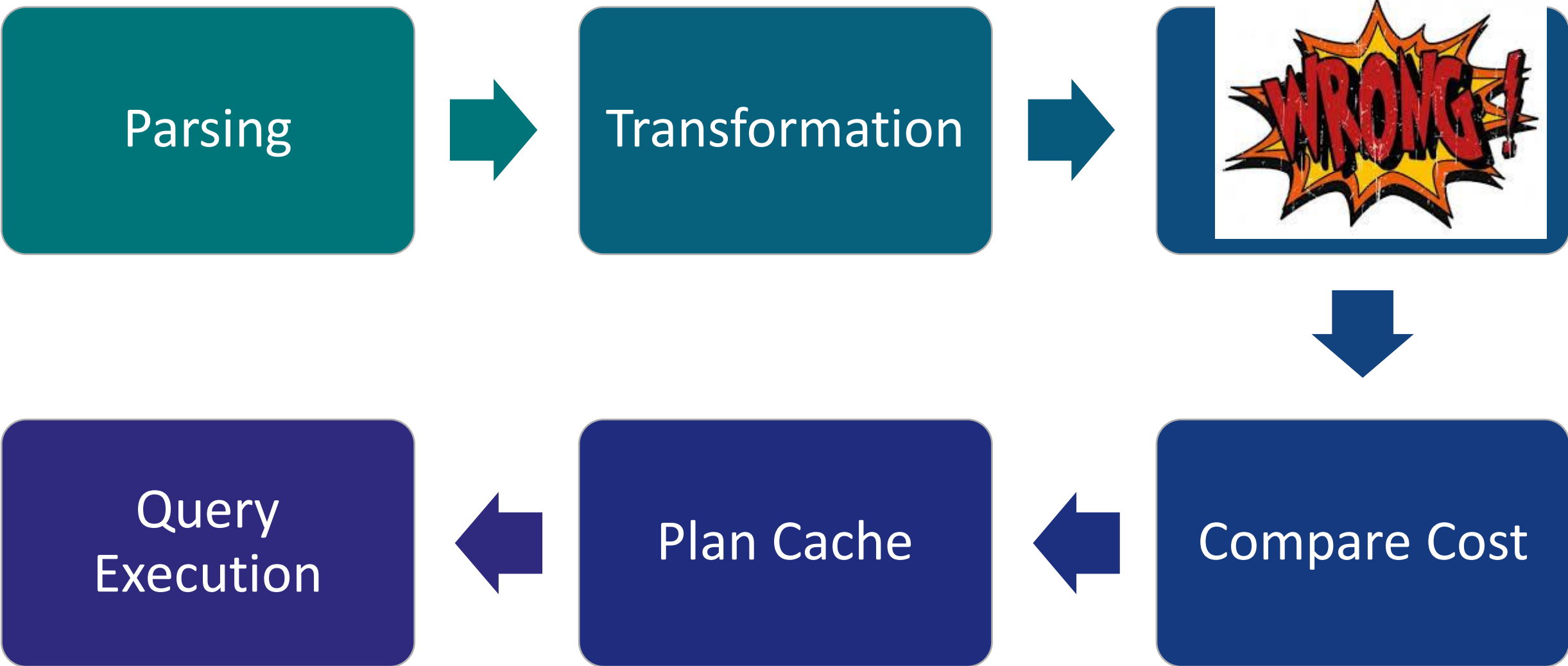
Microsoft Data Platform MVP. 15 Years working with Microsoft Data Platform. Microsoft and MongoDB certified. Worked in ecommerce, healthcare and finance industry.

### Giving Back

Board member NESQL user group and Founder of DBA virtual group. Organizer of Boston Data Saturday. Frequent speaker at local and virtual user groups, Data Saturdays and Azure conferences.

### When Not Working

Running – 1x26.2 and 30+x 13.1, Learning US history. Shuttling 3 kids.



## Cost

Parallel

Serial

## Memory Grant

In  
Memory

Spill to  
Disk

## Access Method

Seek

Scan

Seek +  
Scan

## Algorithm

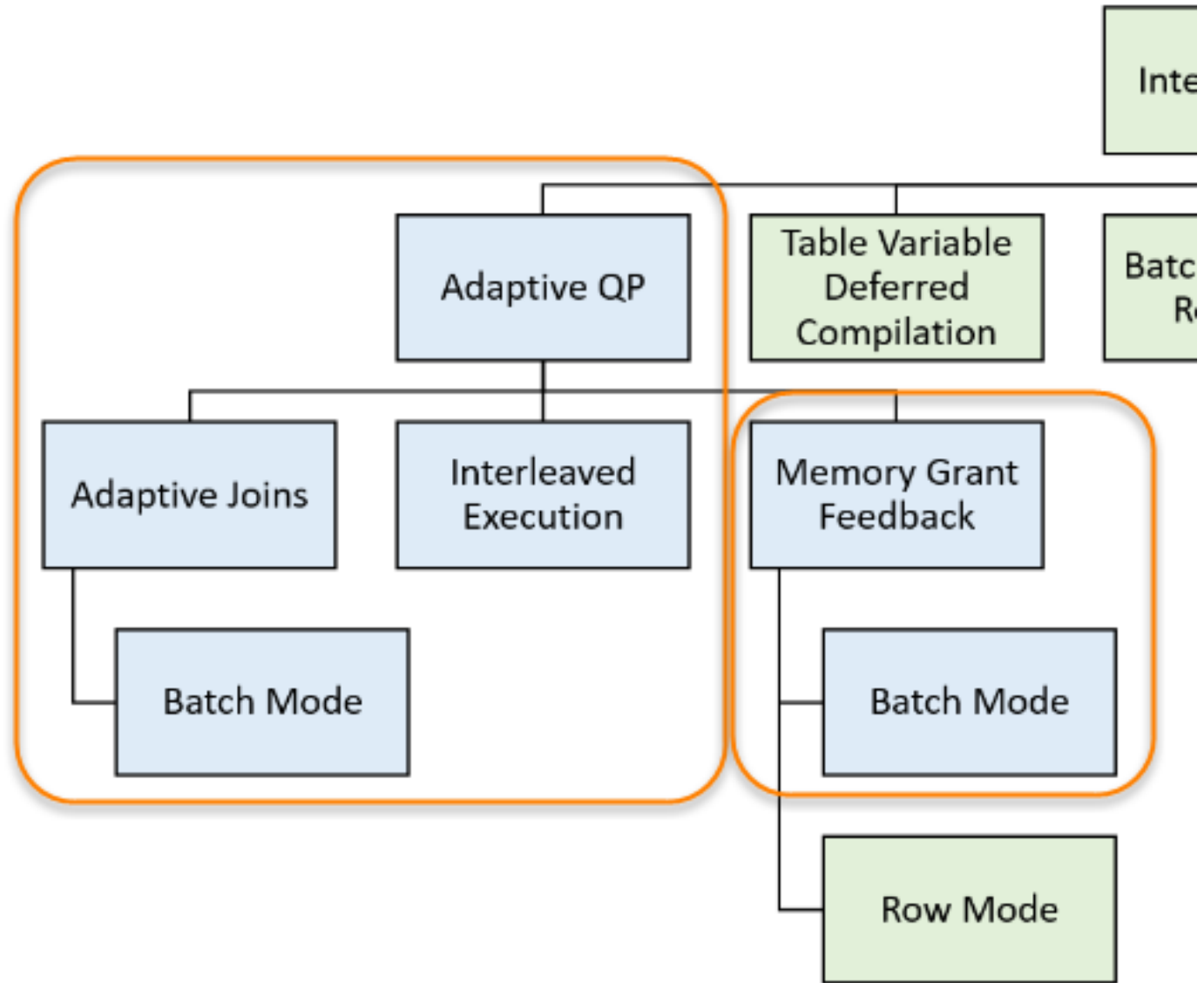
Join

Aggregate

Sort

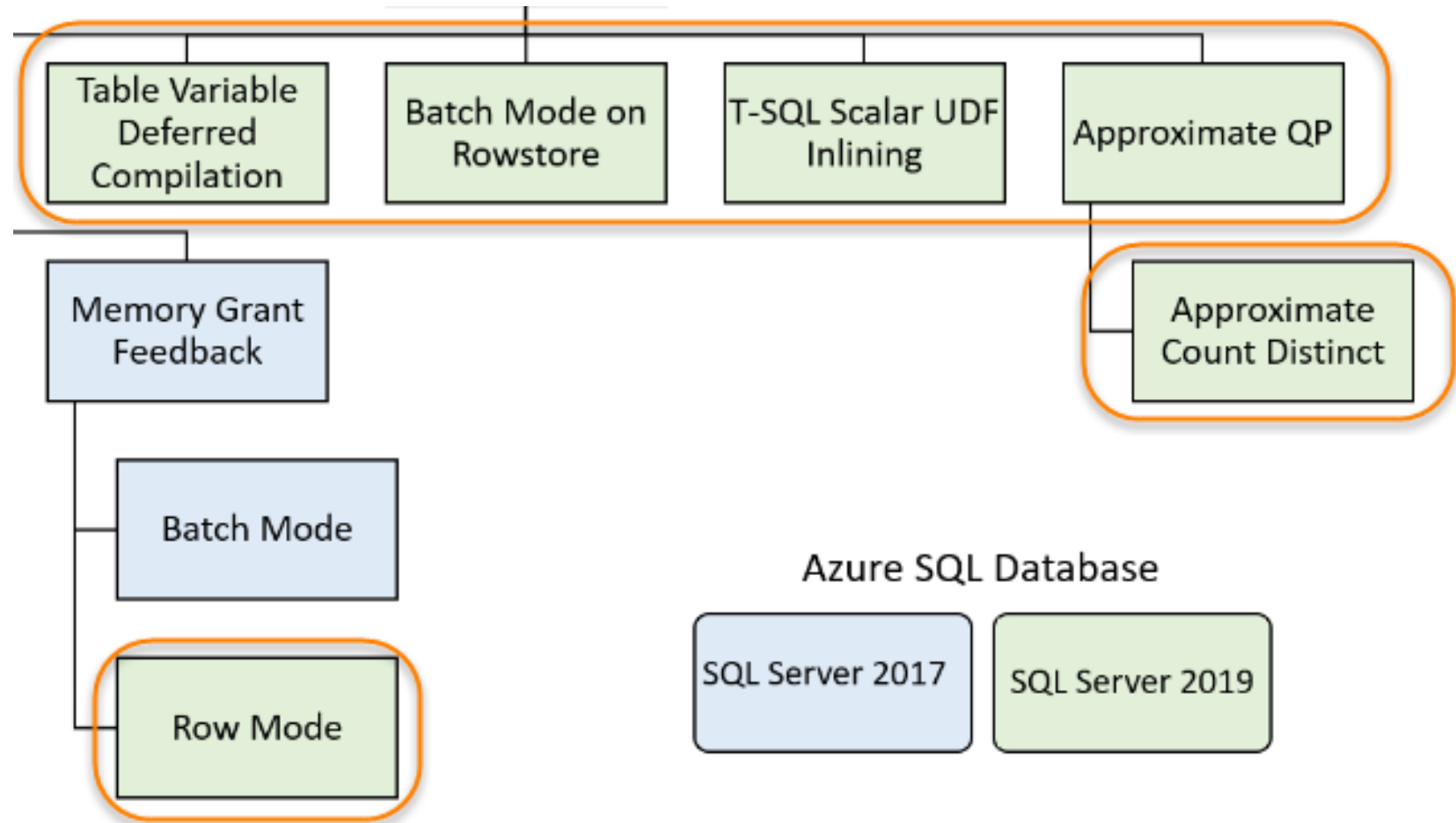


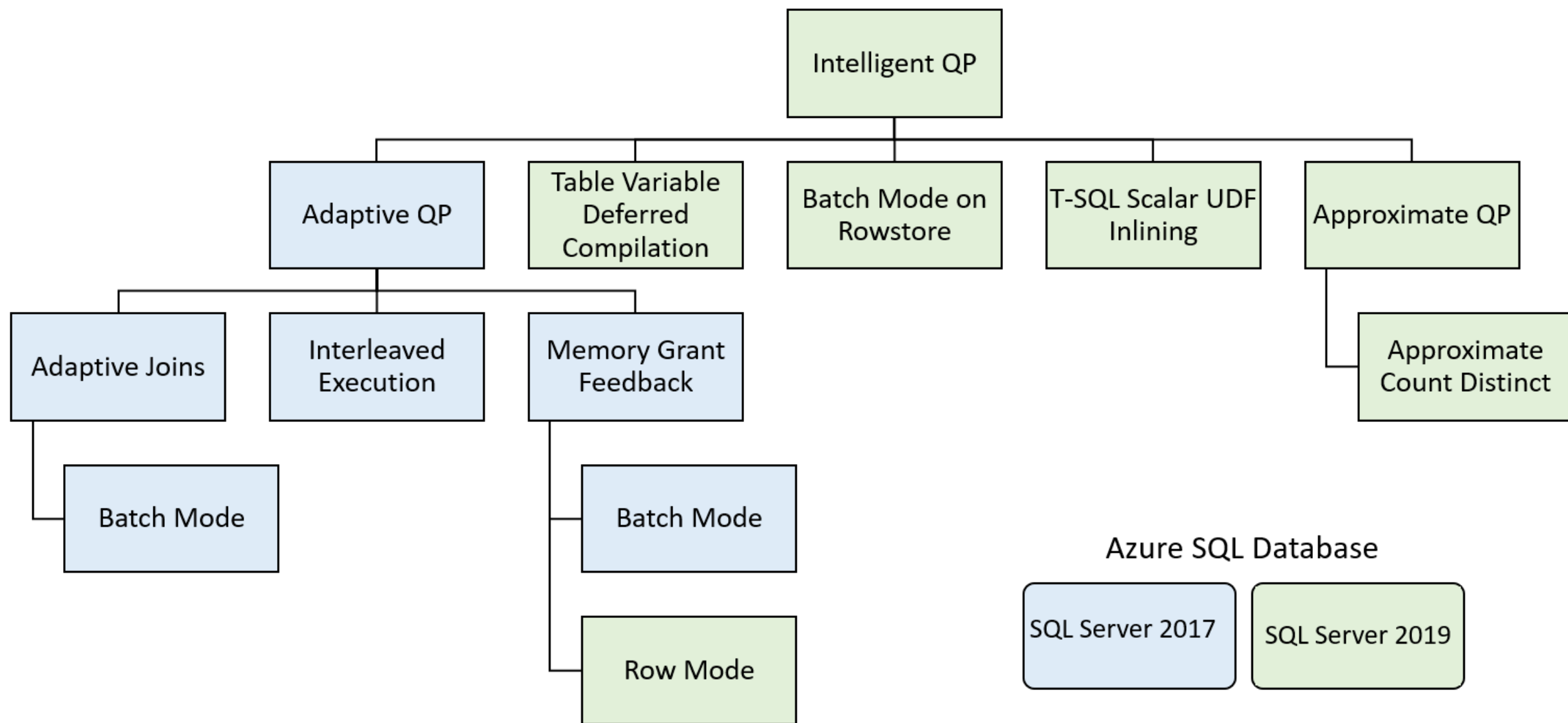
# 2017





# 2019







# 2021

## CE Feedback

- 2021, Preview, Azure SQL Database
- Significant model estimation errors
- Only repeating workload will qualify
  - Cache persistent plan
- During conflict hard-coded hint will prevail
- Only verified feedback will prevail
- Example
  - Correlation Analysis
  - Row Goal
  - Containment type





# 2021

## Query Store Hints

- 2021, Preview, Azure SQL Database, Azure SQL Managed Instance
- Add hints when you do not have access to the code
- Using `query_id` (from Query Store)

# SYS.DATABASE\_SCOPED\_ CONFIGURATIONS

**SYS.DM\_EXEC\_VALID\_US  
E\_HINTS**



# Problem



# Solution

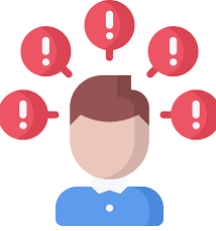


# Caution

## Adaptive Joins Batch Mode

140  
150

- Join Hint
- Parameter sensitive query

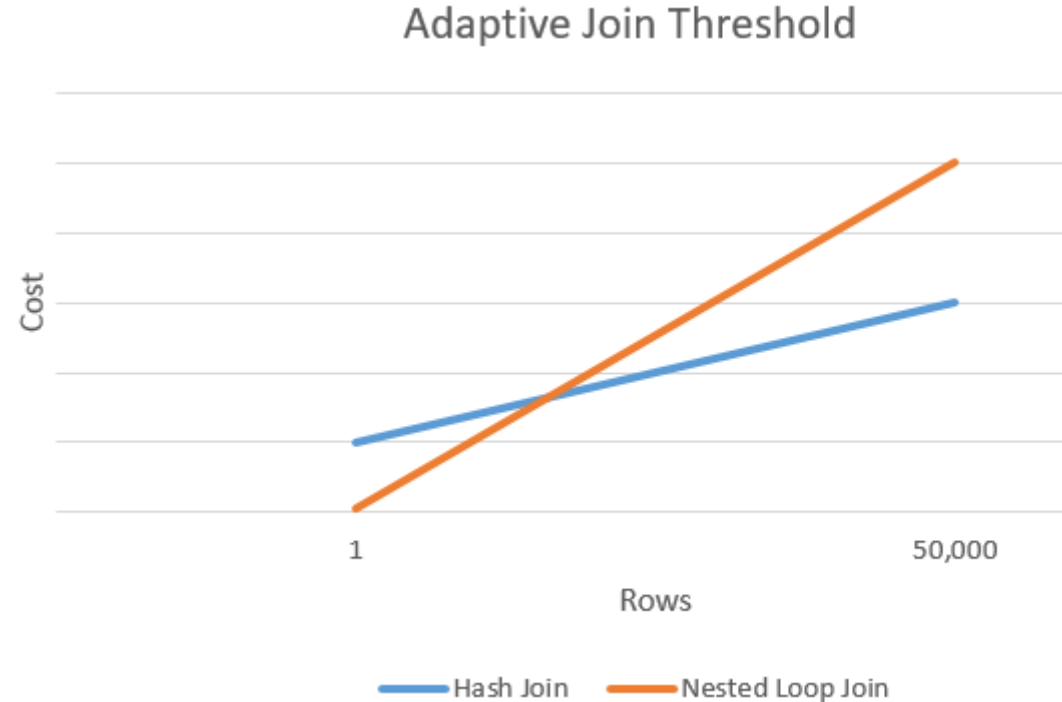




## Adaptive Joins Batch Mode

140  
150

- Dynamically switch to a better join
- Nested loop or Hash Join
- Based on threshold



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

## Adaptive Joins Batch Mode

140  
150

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





## Table Variable Deferred Compilation

150

- Works ok with low number of rows but not as rows increase
- 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 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





## Interleaved Execution MSTVFs

140  
150

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



## Interleaved Execution MSTVFs

140  
150

- Actual row counts are used to make better-informed decision
- Greater performance impact with higher skew



## Interleaved Execution MSTVFs

140  
150

- Must be read-only and NOT part of a data modification
- Must use [runtime constant](#)
- Once an interleaved execution plan is cached, that revised estimate is used for consecutive executions without re-instantiating interleaved execution



## Memory Grant Feedback Batch Mode

140  
150

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



## Memory Grant Feedback Batch Mode

140  
150

- 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  
150

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







## Memory Grant Feedback Row Mode

150

- Row mode memory grant feedback expands on the batch mode



## TSQL Scalar UDF Inlining

150

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



## TSQL 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 inlinng



# TSQL Scalar UDF Inlining

150

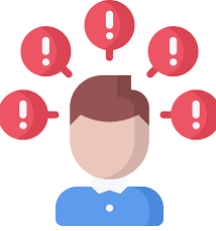
- [Requirements to be eligible](#)
- [Scalar UDF Inlining issues in SQL Server 2019](#)
- [sys.sql\\_modules](#) – Can this UDF be inlined?
- Can disable within function definition

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

## 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 use of Columnstore index



## Batch Mode on Rowstore

150

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



## Batch Mode on Rowstore

150

- Batch mode restriction 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





Approximate  
Count Distinct

150

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





## Approximate Count Distinct

150

- Access of data sets that are millions of rows or higher
- Aggregation of a column or columns that have many distinct values
- Use less memory compared to exhaustive COUNT DISTINCT
- Based on [HyperLogLog](#) algorithm



Approximate  
Count Distinct

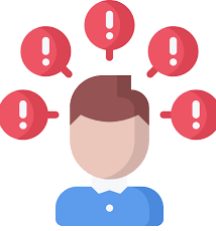
150

- The function implementation guarantees up to a 2% error rate within a 97% probability

## Query Store Hints

# Azure SQL Database

- 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

Azure SQL  
Database

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

# Azure SQL Database



- 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

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;
```

You can also disable any of these features for a specific query by using 'USE HINT' query hint

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

# Resource

- **Intelligent Query Processing in SQL databases**
- **Intelligent Query Processing Demos**
- **Compatibility Certification**
- **Query Processing Architecture Guide**
- **Get Your Scalar UDFs to Run Faster Without Code Changes**
- **Batch Mode Bitmaps in SQL Server by Paul White**
- **Introducing Batch Mode Adaptive Memory Grant Feedback by Joe Sack**
- **Query Store hints (Preview)**



SQL2019 CU13

SSMS 18.10

Azure SQL Database





@sqlworldwide



linkedin.com/in/sqlworldwide



sqlworldwide.com



taio@sqlworldwide.com

