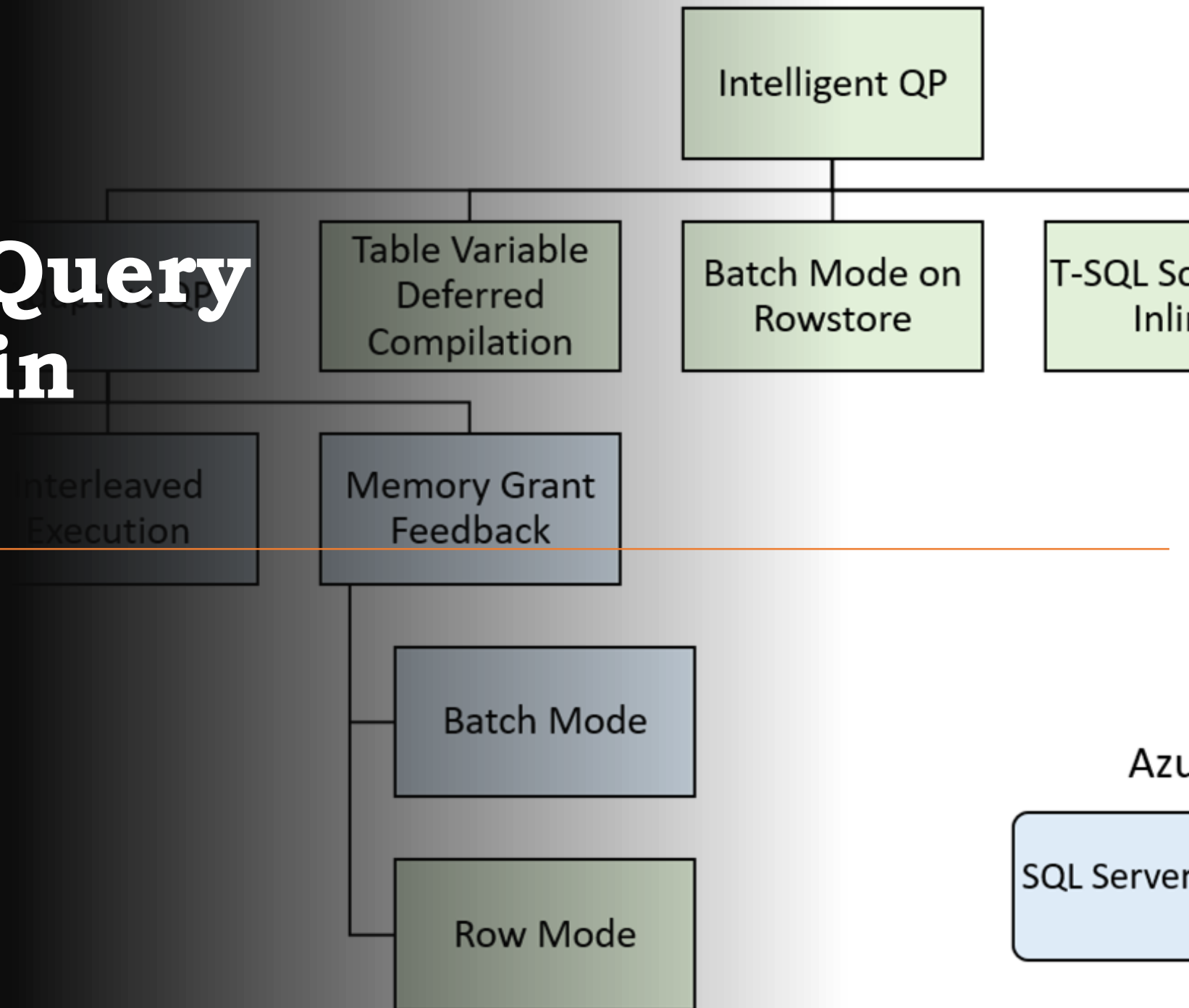


# Intelligent Query Processing in SQL Server

Taib Ali





## Data Solutions Manager, GMO LLC



<http://sqlworldwide.com/>



/sqlworldwide



@sqlworldwide



Taiob@sqlworldwide.com

### Data Professional

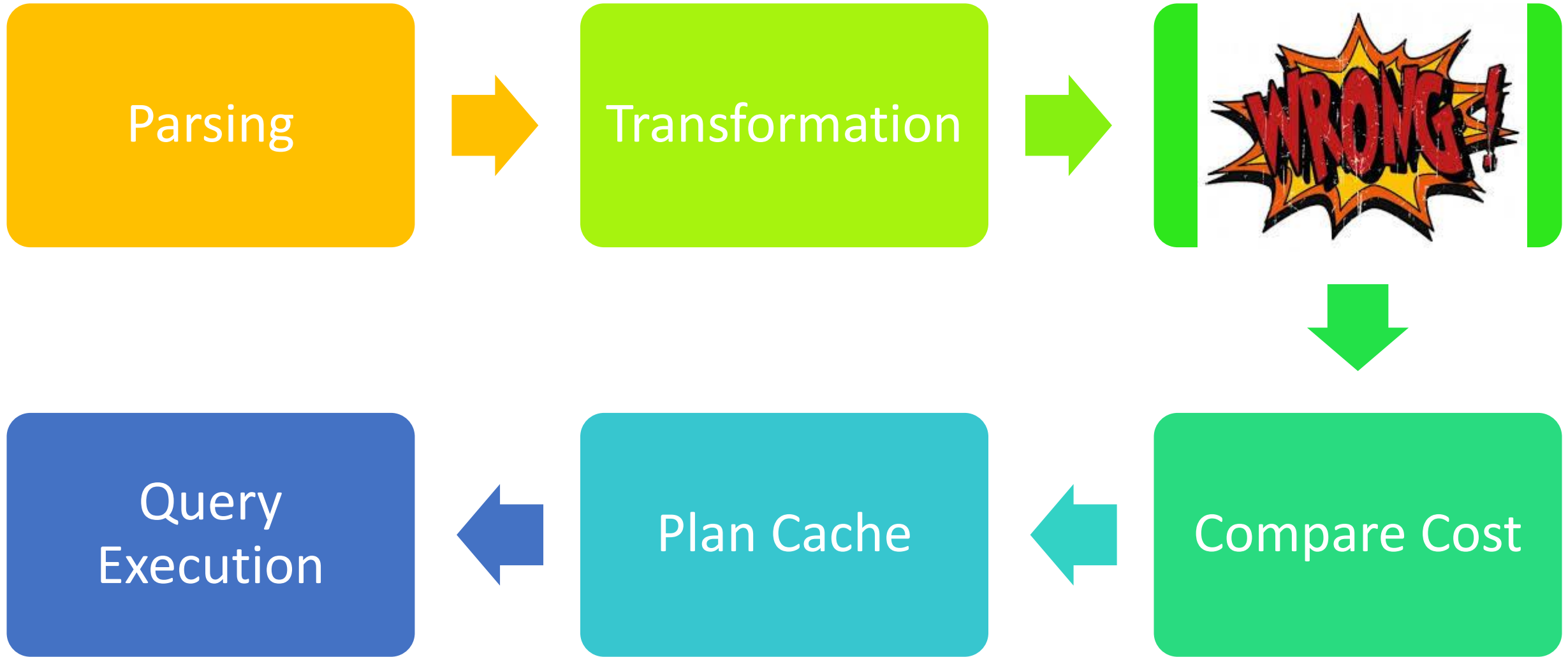
Microsoft Data Platform MVP. 14 Years working with Microsoft Data Platform. Microsoft and MongoDB certified. Worked in ecommerce, healthcare and finance industry.

### Giving Back

Board member NESQL user group and PASS DBA virtual group. Organizer of Boston SQL Saturday. Frequent speaker at local and virtual user groups, SQL Saturdays, and azure events.

### When Not Working

Running – One 26.2 and many 13.1 miles. Learning US history. Shuttling 3 kids.



## Cost

Parallel

Serial

## Memory Grant

In  
Memory

Spill to  
Disk

## Access Method

Seek

Scan

Seek +  
Scan

## Algorithm

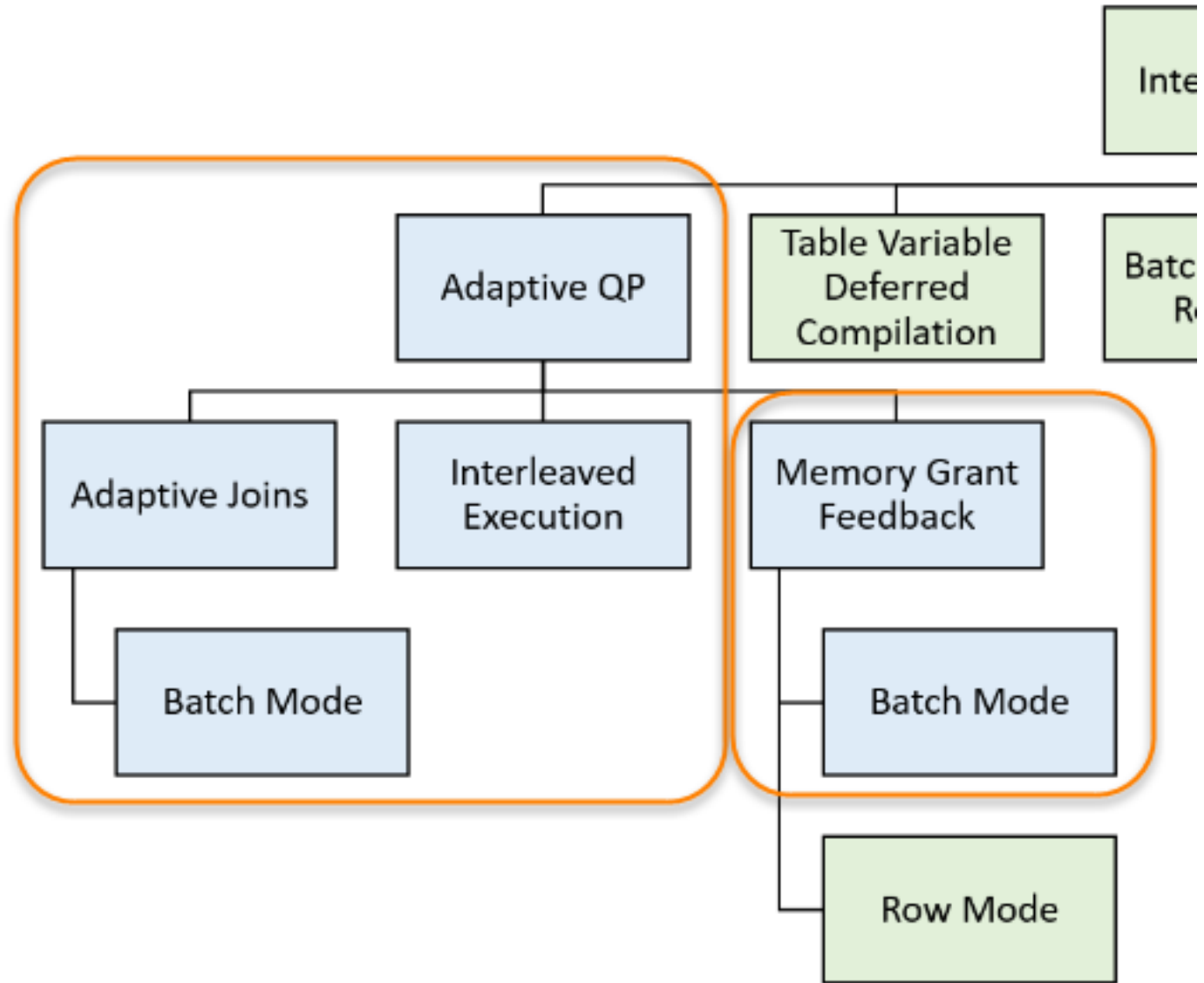
Join

Aggregate

Sort

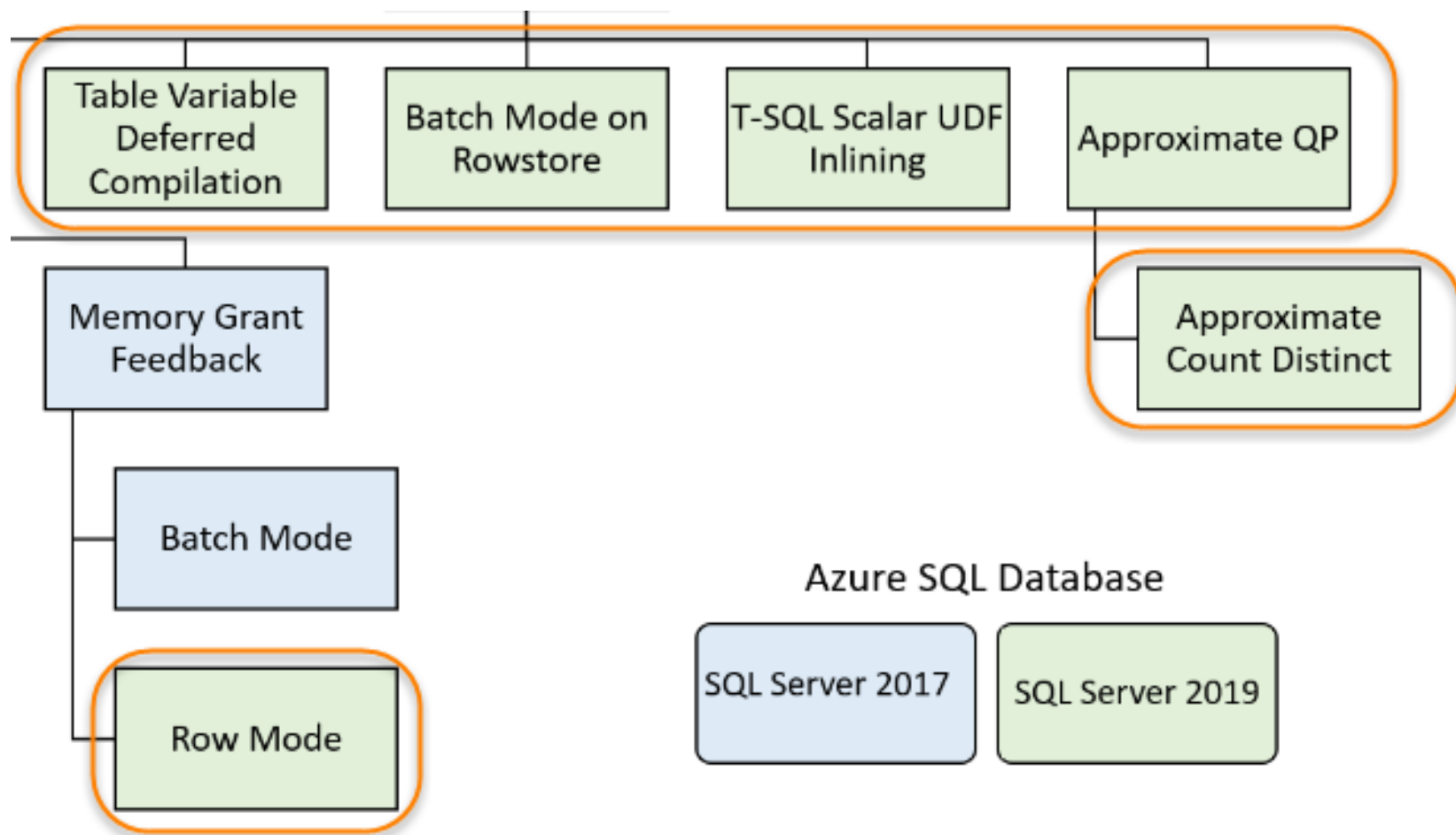


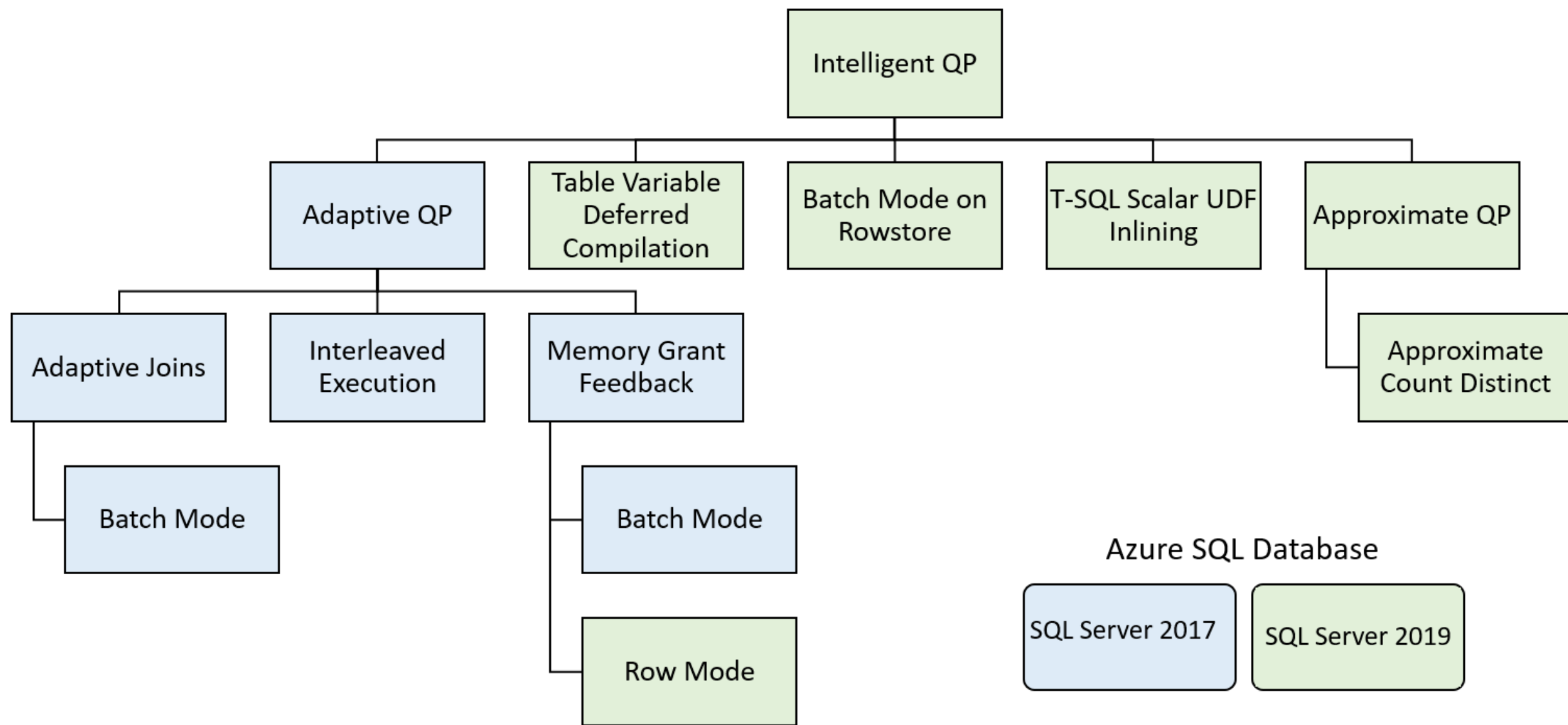
# 2017





# 2019





# SYS.DATABASE\_SCOPED\_ CONFIGURATIONS



**SYS.DM\_EXEC\_VALID\_USE\_HINTS**  
**23 as of SQL2019 CU5**



# Problem



# Solution



# Caution



## Adaptive Joins Batch Mode

140  
150

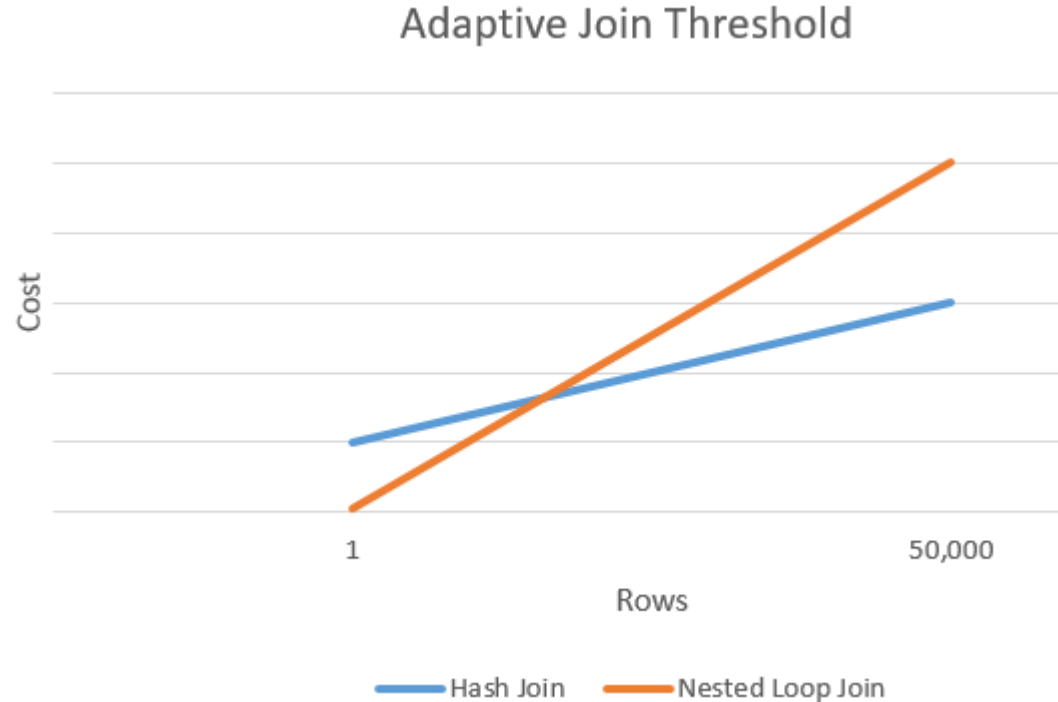
- Join Hint
- Parameter sensitive query



## 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\\_aqjointhreshold.png?view=sql-server-ver15](https://docs.microsoft.com/en-us/sql/relational-databases/performance/media/6_aqjointhreshold.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 join should have same outer reference





## 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 only
- 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 Parament 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](#)



## 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 compare 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



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'));
```



SQL2019 CU5  
SSMS 18.5.1

# 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**



@SqlWorldWide



[linkedin.com/in/sqlworldwide](https://linkedin.com/in/sqlworldwide)



[sqlworldwide.com](https://sqlworldwide.com)



[taio@sqlworldwide.com](mailto:taio@sqlworldwide.com)

