SQL Server 2012:
Nonclustered Columnstore Indexes

Module 3: Leveraging Batch-Execution Mode

Joe Sack

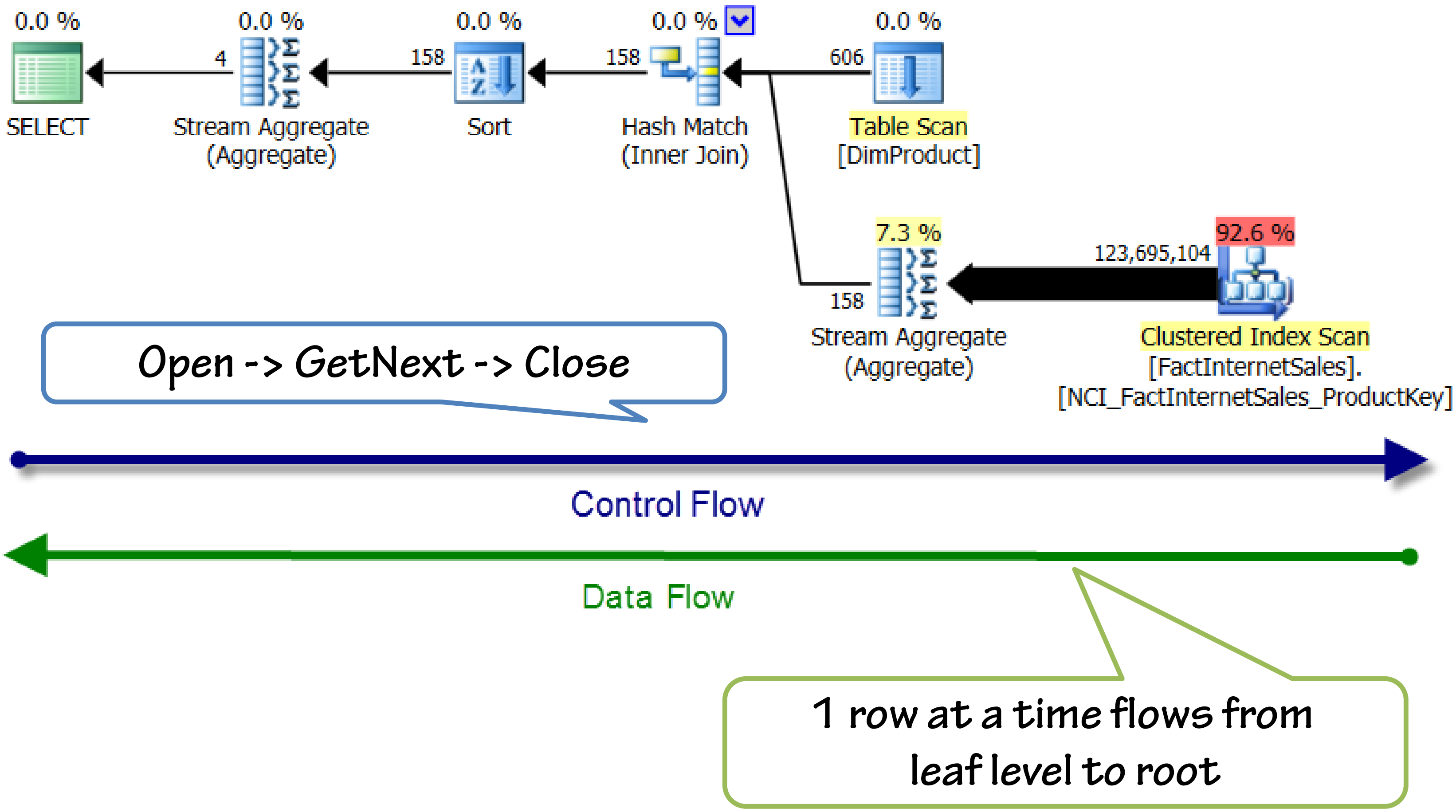
Joe@SQLskills.com

pluralsight
hardcore developer training

Module Introduction

- Batch execution mode allows for optimized execution and is a major component in improving overall query execution time in combination with nonclustered columnstore indexes
- While some workloads may "just work" by automatically using batch execution mode, some will not
- This module will explore batch execution mode and teach you how to identify and address batch execution mode "inhibitors"

Traditional Row Execution Mode



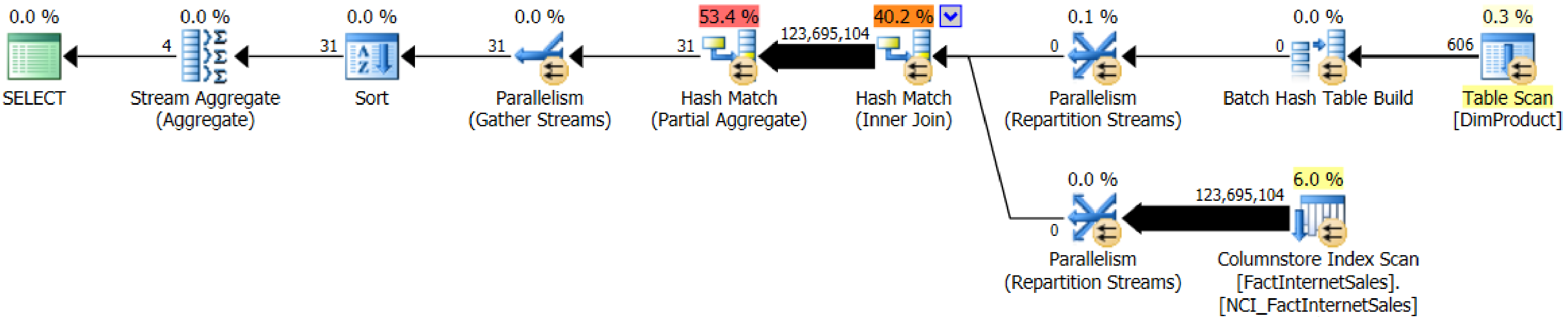
Open -> GetNext -> Close

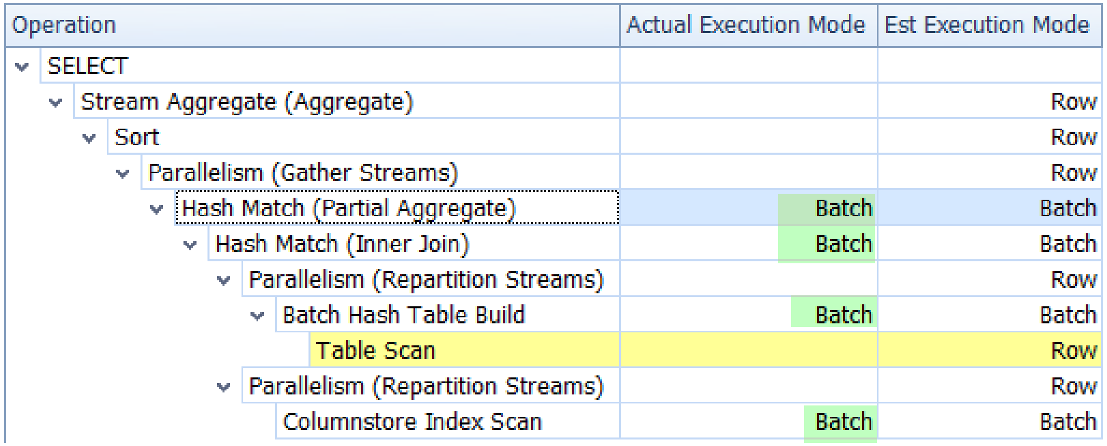
Control Flow

Data Flow

1 row at a time flows from leaf level to root

Batch Execution Mode





Operation	Actual Execution Mode	Est Execution Mode
SELECT		
Stream Aggregate (Aggregate)		Row
Sort		Row
Parallelism (Gather Streams)		Row
Hash Match (Partial Aggregate)	Batch	Batch
Hash Match (Inner Join)	Batch	Batch
Parallelism (Repartition Streams)		Row
Batch Hash Table Build	Batch	Batch
Table Scan		Row
Parallelism (Repartition Streams)		Row
Columnstore Index Scan	Batch	Batch

Batches of rows pushing up the tree for efficient processing

Batch Mode Operator Support

- Columnstore Index Scan
- Hash operations (aggregate, join, batch hash table build)
- Filter
- Compute Scalar

Batch Execution Inhibitors (1)

- **In-memory hash tables not fitting into memory**
  - Cardinality estimate issues
    - For an in-depth review of cardinality estimate issues, see the Pluralsight course “SQL Server: Troubleshooting Query Plan Quality Issues” (http://bit.ly/WRwSpD)
  - Detectable via batch_hash_table_build_bailout XE event

- **Not enough threads for parallel execution**
  - Row execution mode
  - Symptom -> serial execution

Batch Execution Inhibitors (2)

- OUTER JOIN for columnstore-indexed table
- IN, EXISTS, NOT IN, NOT EXISTS
- UNION ALL with columnstore-indexed and row-store tables
- 2+ DISTINCT aggregates
- Scalar aggregates