SQLServerFast.com Execution Plan Video Training

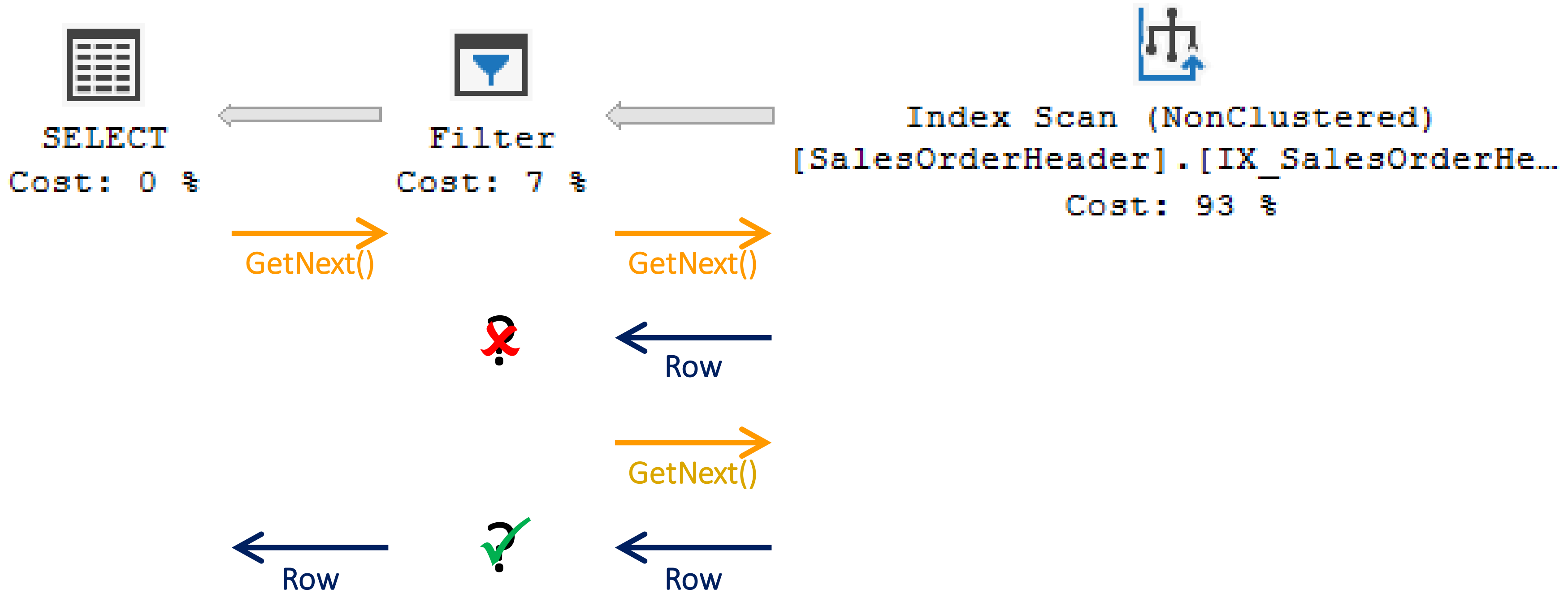
Block 1: Understanding execution plans

Level: Advanced

Chapter 5: Batch mode versus row mode

Batch mode versus row mode

Row mode execution

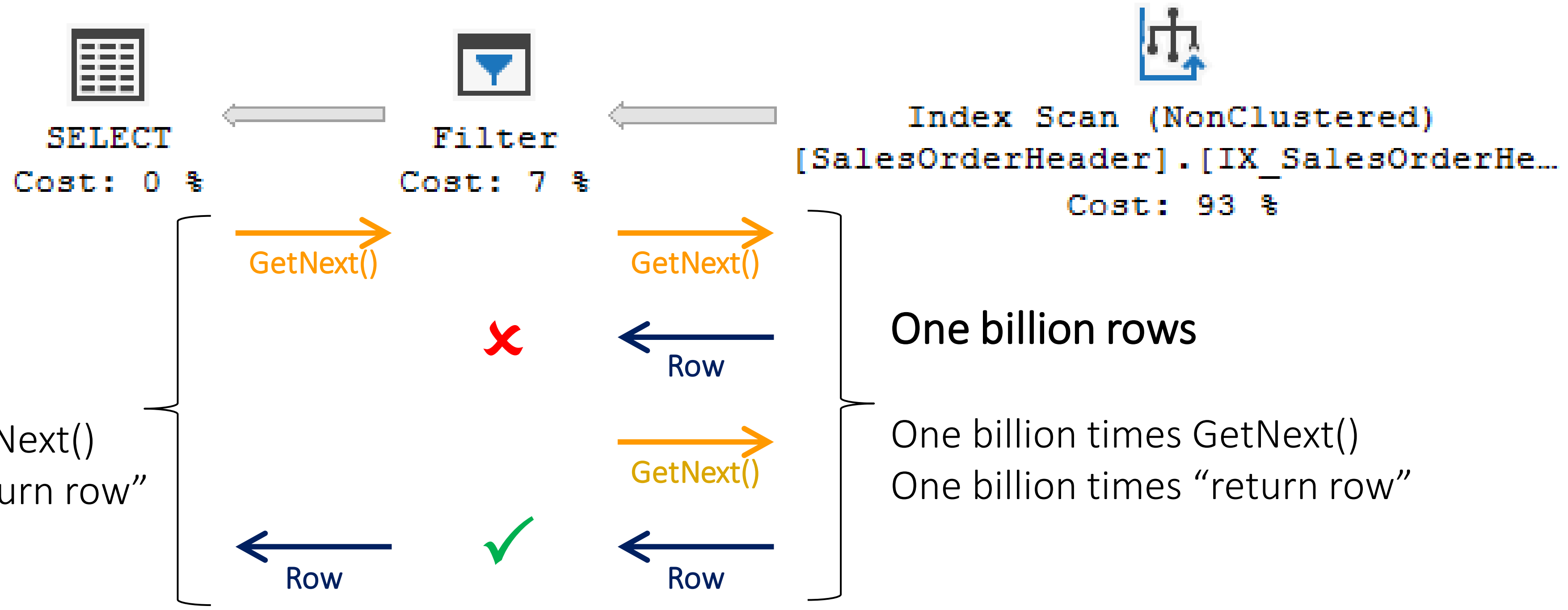


Batch mode versus row mode

- Row mode execution
  - Every call and every return is a pass of control
    - Store current state
    - Load instructions of called operator in instruction cache
    - Execute code
    - Reload instructions of calling operator in instruction cache
    - Restore saved state

Batch mode versus row mode

Row mode execution



500 M matches

500 M times GetNext()
500 M times "return row"

One billion rows

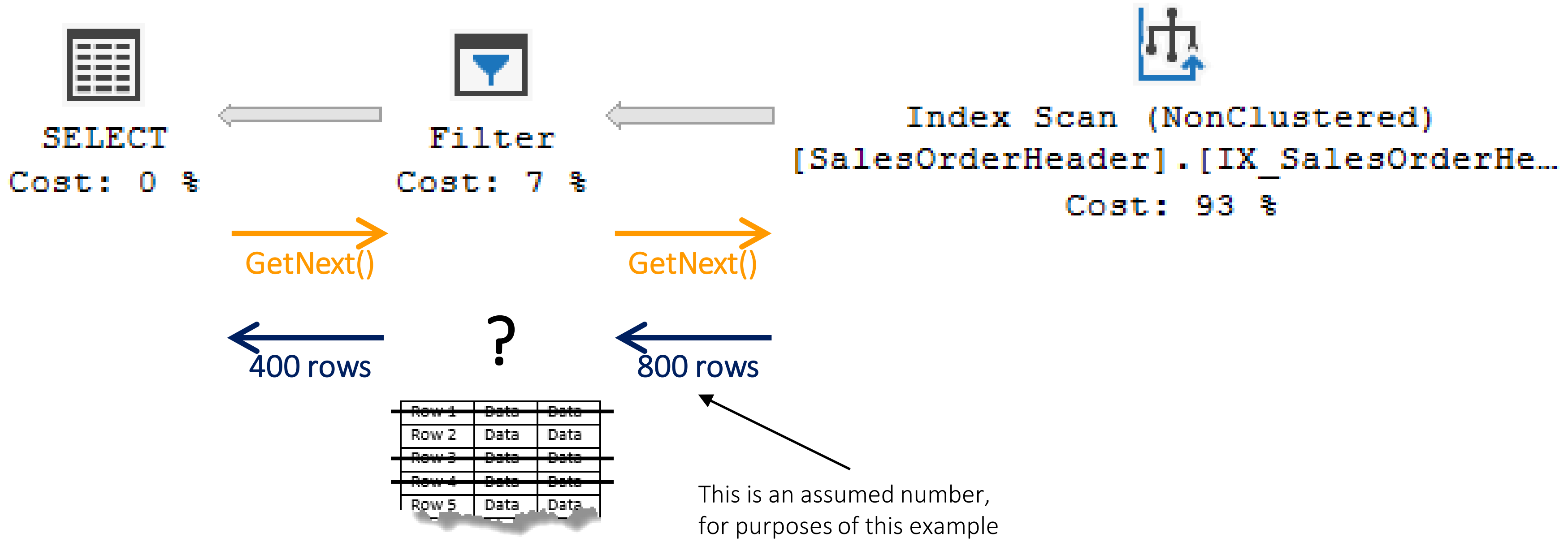
One billion times GetNext()
One billion times "return row"

Batch mode versus row mode

- Row mode execution
  - Every call is a pass of control
    - Store current state
    - Load instructions of called operator in instruction cache
    - Execute code
    - Reload instructions of calling operator in instruction cache
    - Restore saved state
  - High number of passes of control affects performance!

Batch mode versus row mode

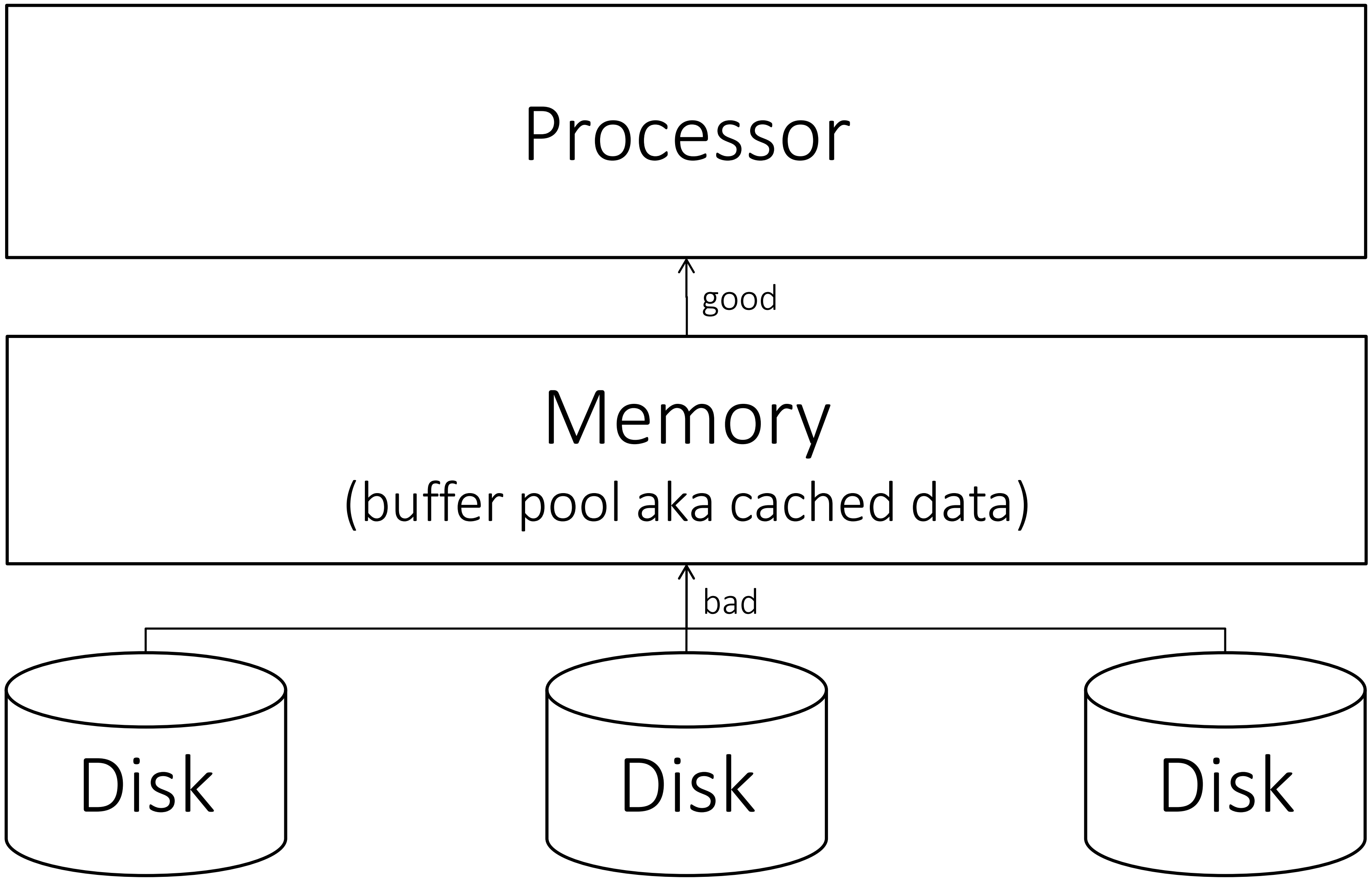
Batch mode execution

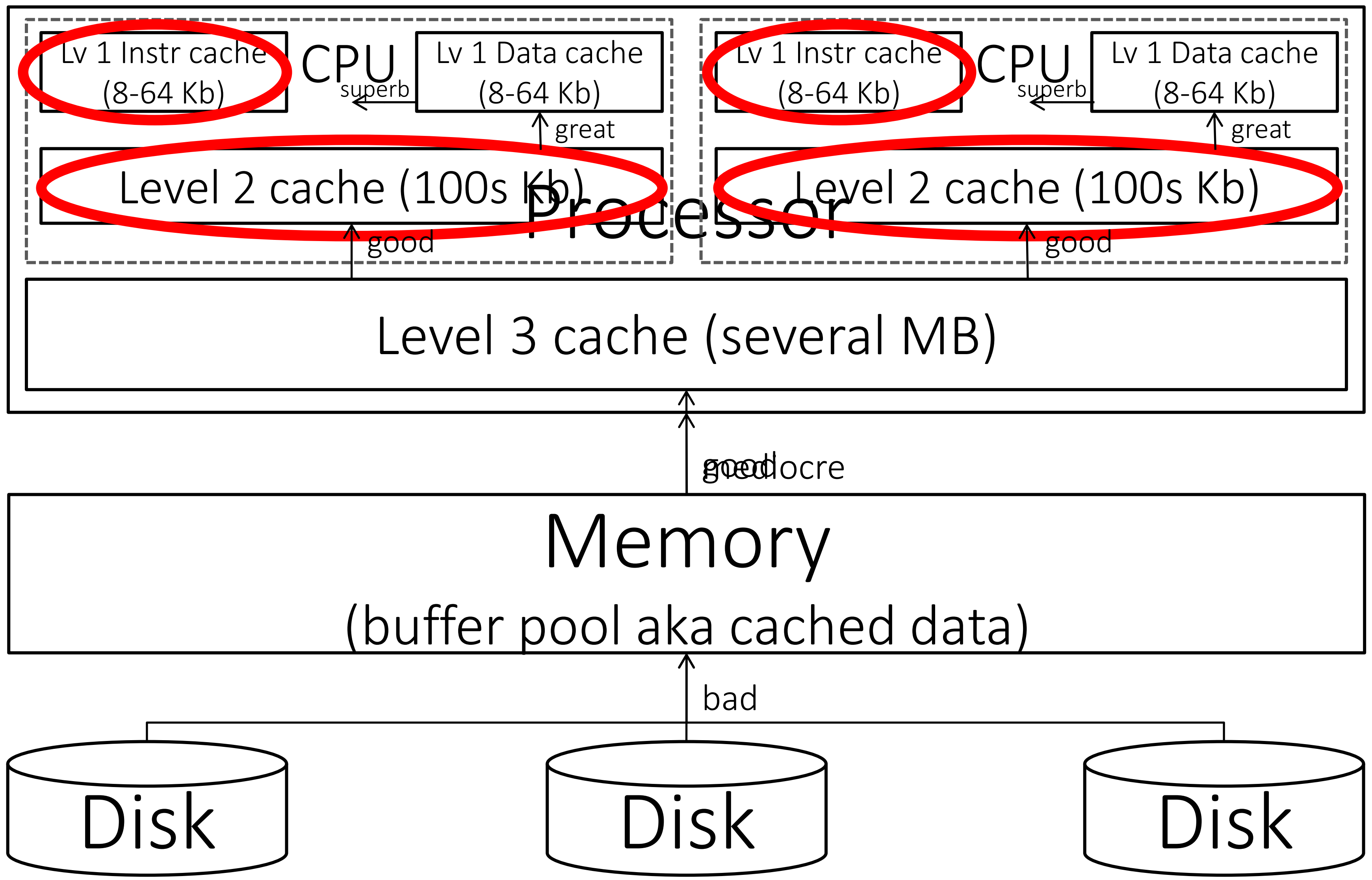


This is an assumed number,
for purposes of this example

Batch mode versus row mode

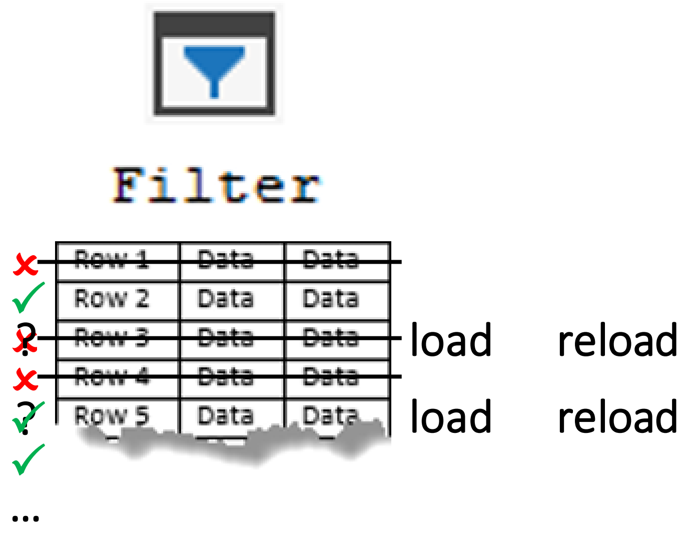
- One billion rows, 500 million matches
  - Row mode: 3 billion passes of control
  - Batch mode: 5 million passes of control
    - *(Based on an assumed batch size of 800 rows → 1 billion / 800 = 1.25 million batches)*
    - Reduced by a factor 600!
  - Why not use batches of a million rows each?
    - Would reduce to just 1000 batches; 4000 passes of control!


Processor
good
Memory
(buffer pool aka cached data)
bad
Disk
Disk
Disk

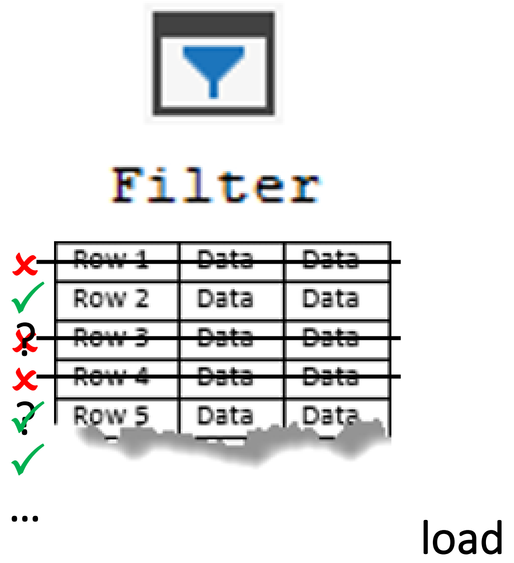

Lv 1 Instr cache (8-64 Kb)
CPU
superb
Lv 1 Data cache (8-64 Kb)
great
Level 2 cache (100s Kb)
good
Processor
Lv 1 Instr cache (8-64 Kb)
CPU
superb
Lv 1 Data cache (8-64 Kb)
great
Level 2 cache (100s Kb)
good
Level 3 cache (several MB)
good
mediocre
Memory
(buffer pool aka cached data)
bad
Disk
Disk
Disk

Optimal level 1 instruction cache usage

Less efficient



More efficient



Batch mode versus row mode

- Performance effect
  - Batch mode can be from 2x to over 80x faster than row mode
  - But there is an overhead
    - Converting row mode (or columnstore) data to batch “vectors”
    - Converting batch vectors back to row mode representation
  - Row mode is still fastest when less data is processed
  - Batch mode can be (much!) faster for large amounts of data

Batch mode versus row mode

- Repacking batches
  - Filter "marks" non-qualifying rows as deleted; they stay in the batch
    - (This uses a bit in the "qualifying rows bitmap")
  - Why not combine qualifying rows in fewer batches?
    - Less batches means less passes of control
    - But: more, and more complex, coding needed
    - Partial batch must be stored somewhere as new batch is passed in
      - The new batch fills the level 2 cache

Batch mode versus row mode

- Batch-preserving operators
  - Rows are marked deleted but not actually removed
  - New columns are stored in pre-allocated space
  - Actual Number of Batches will not change
- Repacking operators
  - Operator creates new batches
  - Rows previously marked deleted are actually removed

Batch mode versus row mode

- Repacking operators (examples)
  - Hash Match (Aggregate)
    - Very high reduction in number of rows
    - Internal implementation doesn’t allow batch-preserving
  - Hash Match (various join operations): sometimes
    - Batch-preserving for one to many relationship
    - Repacking for many to many relationship
      - Impossible to predict number of matches for each input row

Batch mode versus row mode

- Properties
  - Estimated Number of Batches
  - Actual Number of Batches
  - Estimated Execution Mode
    - Determined by optimizer
    - Should the operator run in batch mode or in row mode?
  - Actual Execution Mode
    - Set at run time
    - Did the operator actually execute in batch mode or in row mode?
    - Normally equal to Estimated Execution Mode
      - Exception: SQL Server 2012, when a Hash Match operator spillls to tempdb

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
    - Limited operators
      - Columnstore Index Scan
      - Filter
      - Compute Scalar
      - Hash Match
        - (Inner Join and Aggregate only)
      - Batch Hash Table Build

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
    - Limited operators
    - Parallel execution plan required
    - Query needs to reference at least one table with a columnstore index
    - Limitations often necessitated complex query rewrites
      - Many simple queries had to be rewritten in more complex form
      - Bad for maintainability, good for performance

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
    - Limited operators
      - Columnstore Index Scan
      - Filter
      - Compute Scalar
      - Hash Match
        - (Inner Join and Aggregate only)
      - Batch Hash Table Build
      - Concatenation

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
    - Limited operators
      - Columnstore Index Scan
      - Filter
      - Compute Scalar
      - Hash Match
        - ~~(Inner Join and Aggregate only)~~ (All logical operations)
      - Batch Hash Table Build
      - Concatenation

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
    - Limited operators
      - Columnstore Index Scan
      - Filter
      - Compute Scalar
      - Hash Match
        - (All logical operations)
      - ~~Batch Hash Table Build~~
      - Concatenation

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
    - Limited operators
      - Small changes, but huge effect
      - Almost all “common” queries use batch mode without rewrite

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
    - Limited operators
    - Support for Hash Match spilling without fallback to row mode
    - Seamless switching between row mode and batch mode
      - Does incur overhead, but can be used when appropriate

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
    - Limited operators
    - Support for Hash Match spilling without fallback to row mode
    - Seamless switching between row mode and batch mode
    - Parallel execution plan required
    - Query needs to reference at least one table with a columnstore index

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
  - SQL Server 2016: much better yet
    - Limited operators
      - Columnstore Index Scan
      - Filter
      - Compute Scalar
      - Hash Match
      - Concatenation
      - Sort
      - Window Aggregate

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
  - SQL Server 2016: much better yet
    - Limited operators
    - Batch mode now also supported in serial execution plans
    - Improved memory management
      - Might request additional memory during execution
      - Reduces spills

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
  - SQL Server 2016: much better yet
  - SQL Server 2017: extending on the framework
    - Supported operators
      - Columnstore Index Scan
      - Compute Scalar
      - Concatenation
      - Window Aggregate
      - Filter
      - Hash Match
      - Sort

Batch mode versus row mode

Restrictions on batch mode

- SQL Server 2012: very limited
- SQL Server 2014: improved
- SQL Server 2016: much better yet
- SQL Server 2017: extending on the framework
  - Supported operators
    - Columnstore Index Scan
    - Compute Scalar
    - Concatenation
    - Window Aggregate
    - Filter
    - Hash Match
    - Sort
    - Adaptive Join

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
  - SQL Server 2016: much better yet
  - SQL Server 2017: extending on the framework
    - Supported operators
    - Memory Grant Feedback
      - Works in batch mode only
      - Adjusts memory grant based on previous execution of the same execution plan

Batch mode versus row mode

- Restrictions on batch mode
  - SQL Server 2012: very limited
  - SQL Server 2014: improved
  - SQL Server 2016: much better yet
  - SQL Server 2017: extending on the framework
  - SQL Server 2019: increasing the reach
    - Memory Grant Feedback
      - Works in batch mode ~~only~~ and in row mode
      - Adjusts memory grant based on previous execution of the same execution plan
    - Batch mode on rowstore: no columnstore index required

Summary

- Batch mode versus row mode
  - Less passes of control by processing many rows at once
  - Optimal usage of level 2 cache and level 1 instruction cache
  - Useful for large data sets; too much overhead for smaller sets
  - Introduced in SQL Server 2012
  - Improved in every version since

Next chapters

- Block 2: Reading data – basic level
  - Rowstore data
    - Storage structures
    - Scan operators
    - Seek operators
    - Lookup operators
  - Special scans

Next chapters

- Block 2: Reading data – basic level
- Block 2: Reading data – advanced level
  - Other storage structures
    - Columnstore indexes
    - Memory-optimized indexes
    - Special indexes
  - Special cases for reading data
    - Parallelism, batch mode