INTELIGENTNO PROCESIRANJE U SQL SERVERU 2019

MILOŠ RADIVOJEVIĆ

DATA PLATFORM MVP

26. DECEMBAR 2020

BATCHMOD ON ROWSTORE

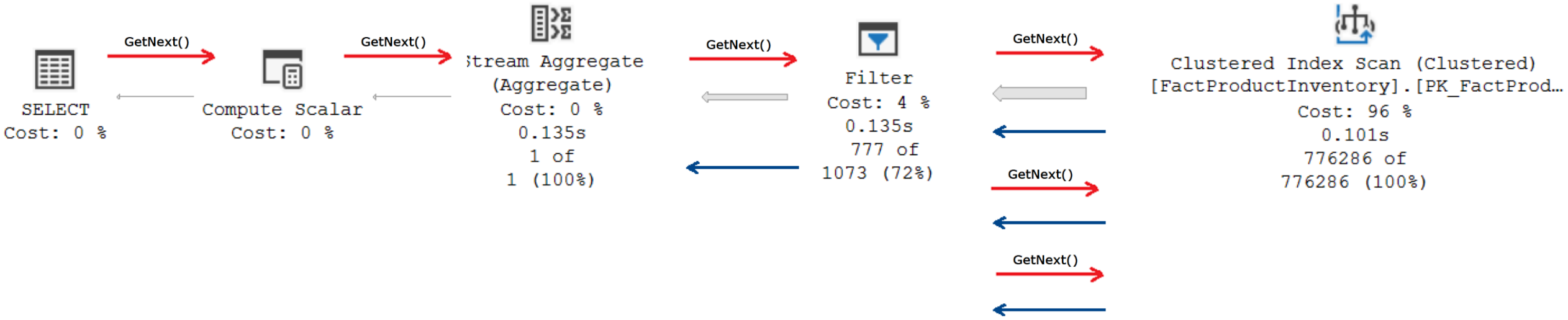
QUERY PROCESSING IN SQL SERVER

- Row mode
  - Efficient and suitable for OLTP scenarios
- Batch mode
  - Suitable for large amounts of data
  - Uses the CPUs more efficiently

BATCH MODE WITH COLUMNSTORE/ROWSTORE

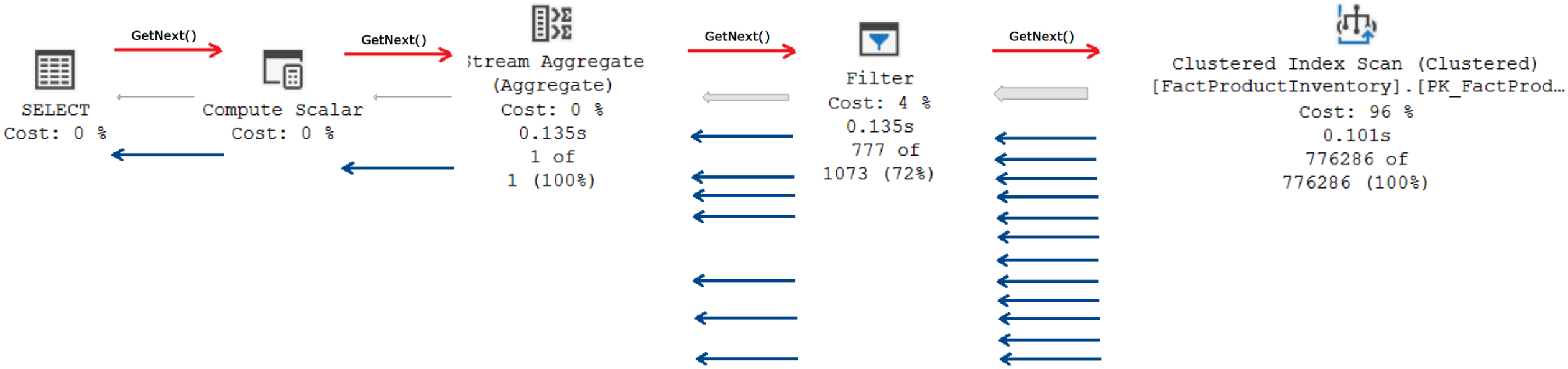
- Batch mode on columnstore was introduced with SQL Server 2012
  - **Improvements - queries up to 20 times faster!**
- Batch mode on rowstore was introduced in SQL Server 2019
  - Some queries can be significantly faster
  - **In my examples 2-5x faster**

ROW MODE



Inefficient; the same instructions for every row, overhead of giving control to another operator and taking it back

BATCH MODE



batch of rows as working unit: up to 900 rows, depends on the number and size of columns and CPU L2 cache

WHAT IS BATCH MODE?

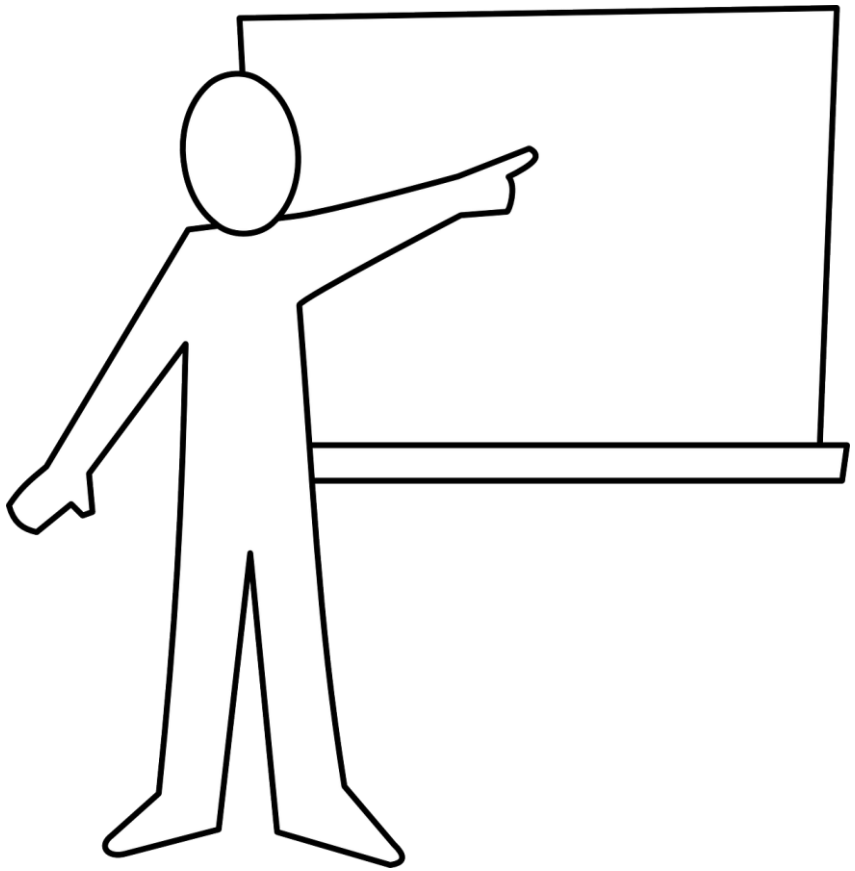
- Batch mode allows query operators to work on a batch of rows, instead of just one row at a time
- At the CPU level multiple rows processed at once instead of one row
- Number of processing instructions reduced
- Better CPU cache utilization and increased memory throughput
- Not exactly documented how to get the number of rows in batch (900 in all my tests)
- Can be beneficial for queries that are CPU bound

BATCH MODE OPERATORS

- Clustered Index Scan
- Table Scan
- Sort
- Hash Match
- Computer Scalar
- Window Aggregate
- Concatenation
- Filter

- Stream Aggregate
- Nested Loops
- Merge Join

BATCH MODE ON ROWSTORE



DEMO

BATCH MODE ON ROWSTORE

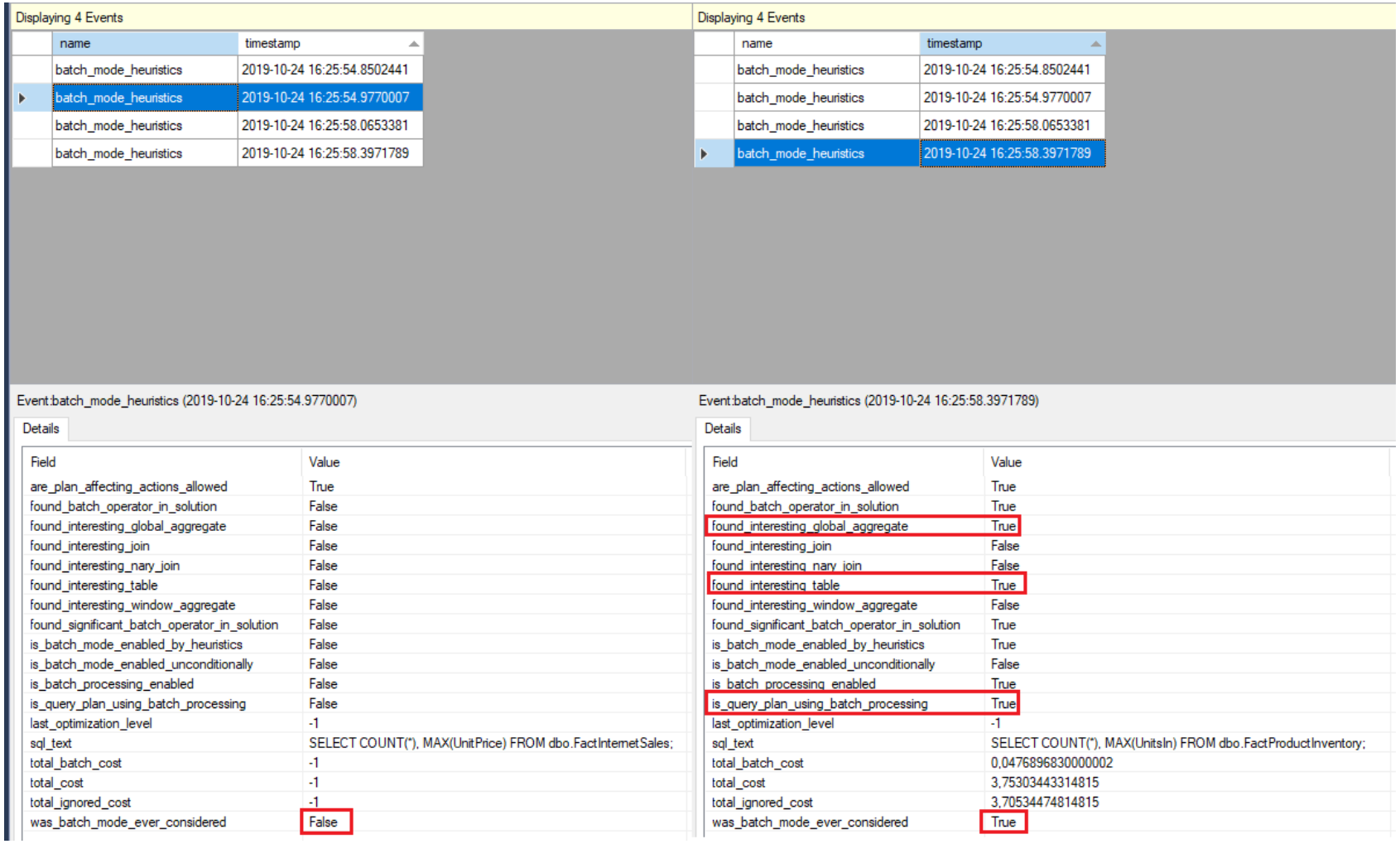
- Initial heuristics considers potential benefits of batch mode for operators
  - Interesting table (at least 131.072 rows)
  - Interesting batch operations: Join, Aggregate or Window Aggregate
  - At least one of the batch operator's input should have not less than 131.072 rows

- Native support
  - No tricks with fake columnstore indexes or other o

UNDOCUMENTED

BATCH MODE ON ROWSTORE

sqlserver.batch_mode_heuristics Extended Event



Displaying 4 Events

name	timestamp
batch_mode_heuristics	2019-10-24 16:25:54.8502441
batch_mode_heuristics	2019-10-24 16:25:54.9770007
batch_mode_heuristics	2019-10-24 16:25:58.0653381
batch_mode_heuristics	2019-10-24 16:25:58.3971789

Event:batch_mode_heuristics (2019-10-24 16:25:54.9770007)

Details

Field	Value
are_plan_affecting_actions_allowed	True
found_batch_operator_in_solution	False
found_interesting_global_aggregate	False
found_interesting_join	False
found_interesting_nary_join	False
found_interesting_table	False
found_interesting_window_aggregate	False
found_significant_batch_operator_in_solution	False
is_batch_mode_enabled_by_heuristics	False
is_batch_mode_enabled_unconditionally	False
is_batch_processing_enabled	False
is_query_plan_using_batch_processing	False
last_optimization_level	-1
sql_text	SELECT COUNT(*), MAX(UnitPrice) FROM dbo.FactInternetSales;
total_batch_cost	-1
total_cost	-1
total_ignored_cost	-1
was_batch_mode_ever_considered	False

Displaying 4 Events

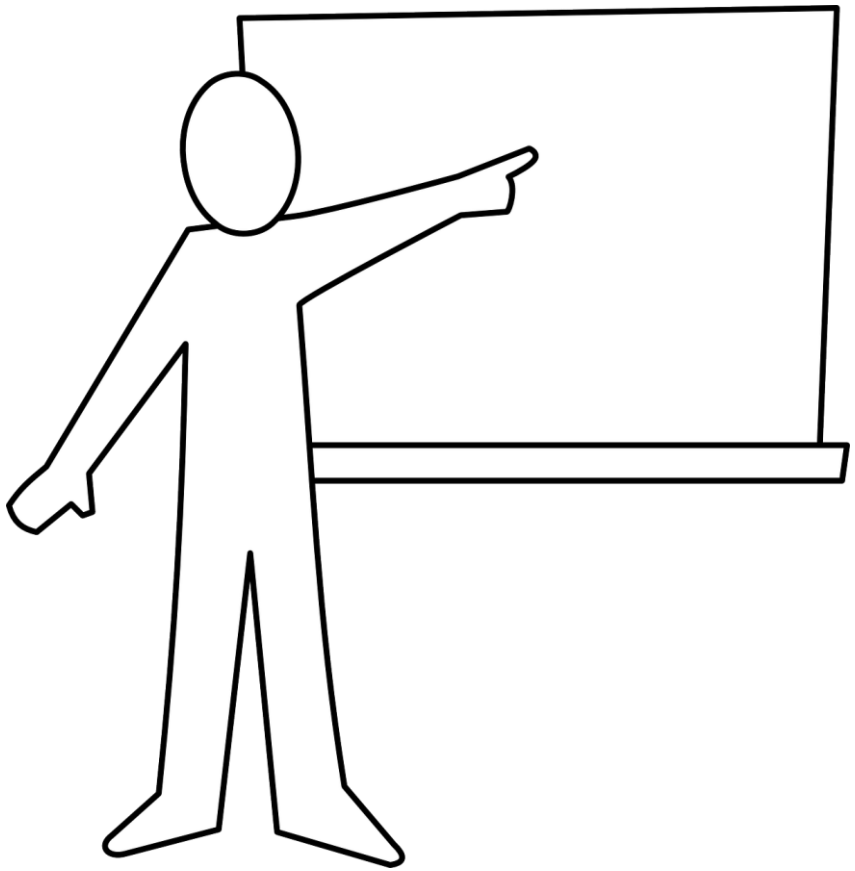
name	timestamp
batch_mode_heuristics	2019-10-24 16:25:54.8502441
batch_mode_heuristics	2019-10-24 16:25:54.9770007
batch_mode_heuristics	2019-10-24 16:25:58.0653381
batch_mode_heuristics	2019-10-24 16:25:58.3971789

Event:batch_mode_heuristics (2019-10-24 16:25:58.3971789)

Details

Field	Value
are_plan_affecting_actions_allowed	True
found_batch_operator_in_solution	True
found_interesting_global_aggregate	True
found_interesting_join	False
found_interesting_nary_join	False
found_interesting_table	True
found_interesting_window_aggregate	False
found_significant_batch_operator_in_solution	True
is_batch_mode_enabled_by_heuristics	True
is_batch_mode_enabled_unconditionally	False
is_batch_processing_enabled	True
is_query_plan_using_batch_processing	True
last_optimization_level	-1
sql_text	SELECT COUNT(*), MAX(UnitsIn) FROM dbo.FactProductInventory;
total_batch_cost	0,0476896830000002
total_cost	3,75303443314815
total_ignored_cost	3,70534474814815
was_batch_mode_ever_considered	True

BATCH MODE ON ROWSTORE



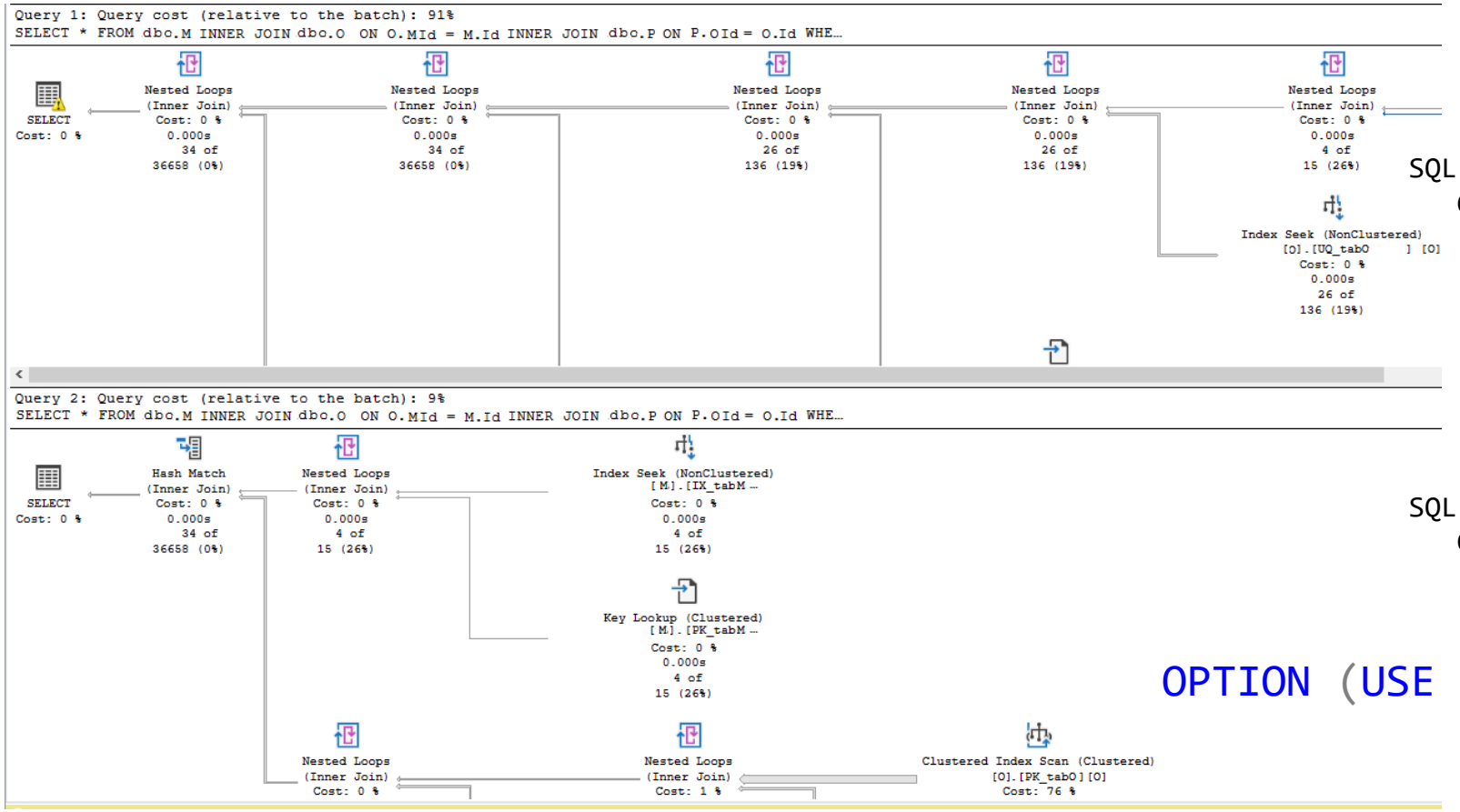
DEMO

```sql
SELECT * FROM dbo.M
INNER JOIN dbo.O ON O.Mid = M.Id
INNER JOIN dbo.P ON P.Oid = O.Id
WHERE M.c1 = 2462782;
```

REGRESSIONS?

25x

```sql
SELECT * FROM dbo.M
INNER JOIN dbo.O ON O.Mid = M.Id
INNER JOIN dbo.P ON P.Oid = O.Id
WHERE M.c1 = 2462782 OPTION (USE HINT ('QUERY_OPTIMIZER_COMPATIBILITY_LEVEL_150'));
```

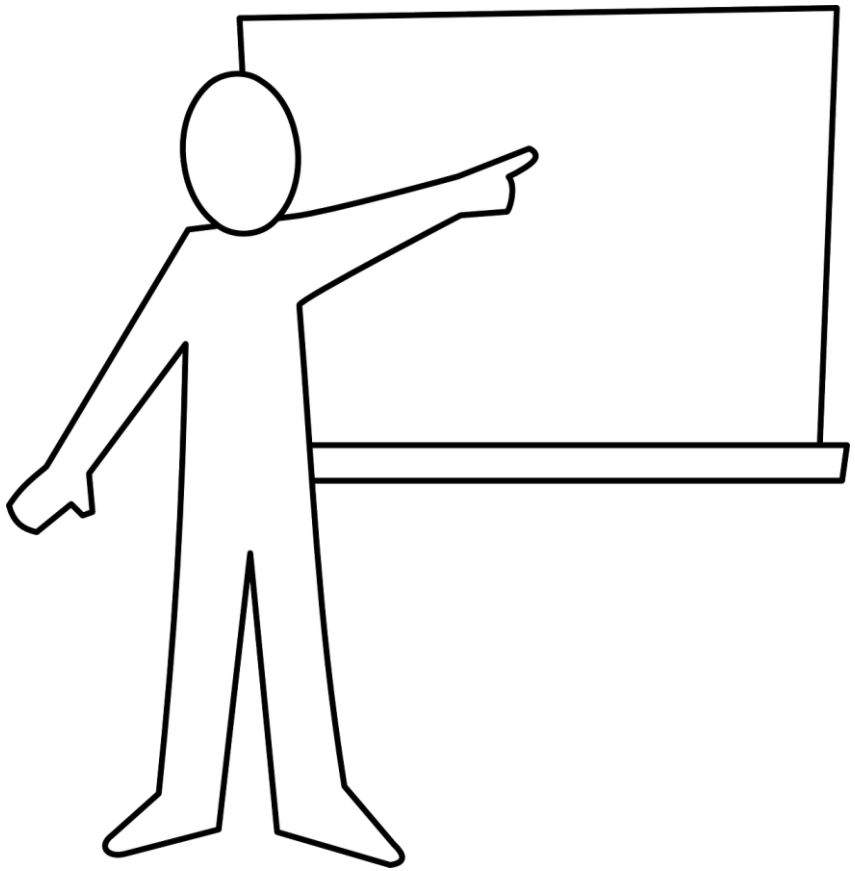


SQL Server Execution Times:
CPU time = 0 ms, elapsed time = **2** ms.

SQL Server Execution Times:
CPU time = 47 ms, elapsed time = **50** ms.

```sql
OPTION (USE HINT('DISALLOW_BATCH_MODE'));
```

BATCH MODE ON ROWSTORE - REGRESSIONS



DEMO

CONFIGURATION

Enable:

```
ALTER DATABASE current SET COMPATIBILITY_LEVEL = 150;

ALTER DATABASE SCOPED CONFIGURATION SET BATCH_MODE_ON_ROWSTORE = ON;

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

Disable:

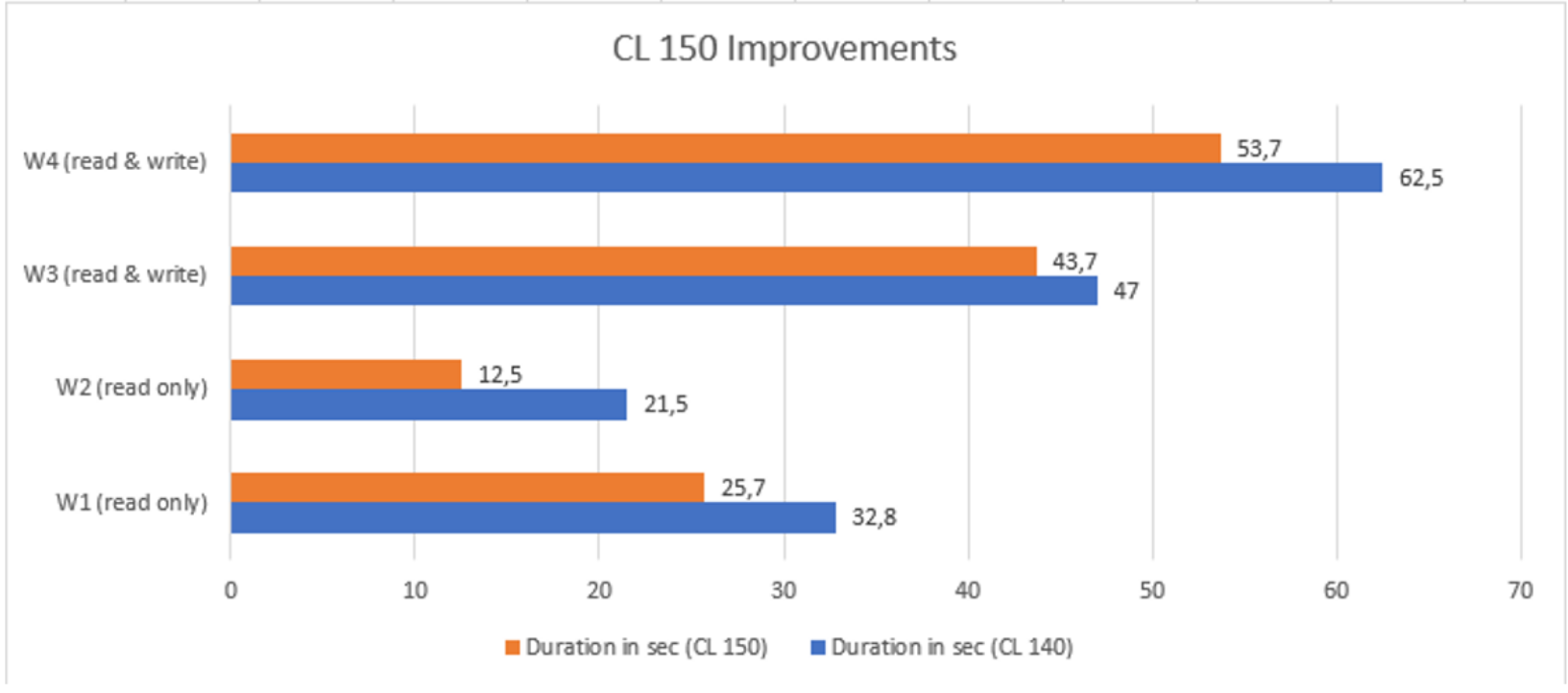
```
ALTER DATABASE SCOPED CONFIGURATION SET BATCH_MODE_ON_ROWSTORE = OFF;

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

UNSERE TESTS – MIT REINEM OLTP WORKLOAD

Workload	Duration in sec (CL 140)	Duration in sec (CL 150)	Improvement
W1 (read only)	32,8	25,7	21,6 %
W2 (read only)	21,5	12,5	41, 8 %
W3 (read & write)	47,0	43,7	7 %
W4 (read & write)	62,5	53,7	14,1 %

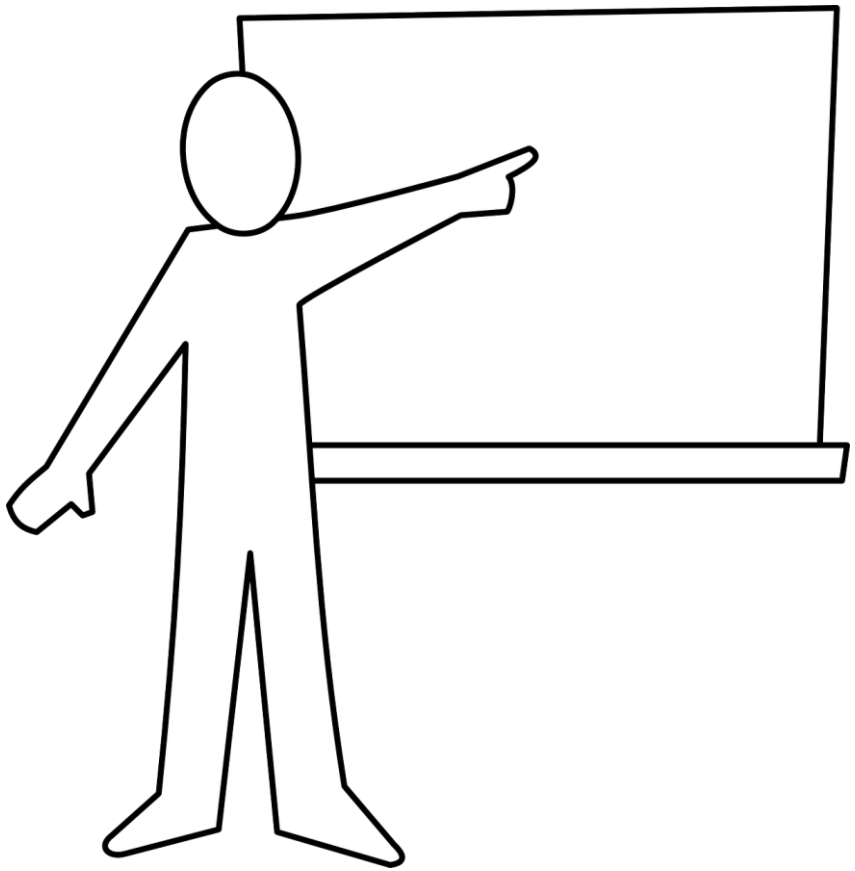
Allgem. Verbesserung – **17%**



LIMITATIONS

- reading from memory-optimized tables will be always done in row mode
- queries use table has (B)LOB, XML or sparse columns in the SELECT or WHERE clause
- queries using full-text or cursors

BATCH MODE ON ROWSTORE - LIMITATIONS



DEMO

CONCLUSION

- Very promising feature
  - Improvements with no efforts
  - It could be a reason for upgrade for some companies
- First version, probably will not optimize all queries, where you would expect the optimization
- Possible regressions, but you can enable/disable feature at two levels
- It brings benefits for queries with large tables and datasets