Niko Neugebauer

Columnstore Indexes Worst Practices & Evil Limitations

Our Partners

Microsoft Data Platform Professional

OH22 (http://www.oh22.net)

Data Platform MVP

Founder of 3 of Portuguese PASS Chapters

Creator of **CISL – Columnstore Indexes Script Library** (https://github.com/NikoNeugebauer/CSIL)

Niko Neugebauer

Blog: http://www.nikoport.com

Twitter: @NikoNeugebauer

LinkedIn: http://pt.linkedin.com/in/webcaravela

Today’s Agenda is Simple

Intro

Overall Recommendation

Building Columnstore Indexes

Loading Data into Columnstore Indexes

Querying the Columnstore Indexes

Recent discoveries

Columnstore Indexes are:

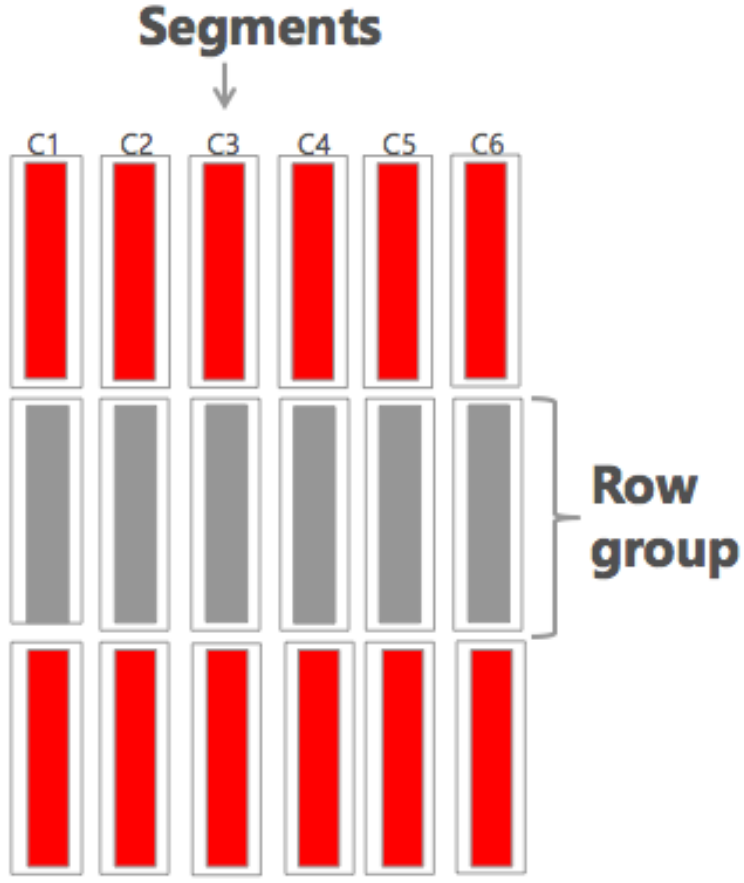
Verytically separated

Grouped into Segments

Extremely compressed

Tuned for processing

large volumes of data



A Row Group CAN contain between 1 and 1048576 rows

The Batch Execution Mode

New Model of Data Processing

Query execution by using **GetNextRow**(), which delivers data to the CPU (In its turn it will go down the stack and get physical accecss to data.
Every operator behaves this way, which makes **GetNextRow** a virtuall function.

For execution plans sometimes you will have 100s of this function invocation before you will get actual 1 row.
For OLTP it might be a good idea, since we are working just with few rows, but If you are working with millions of rows (in BI or DW) you will make billions of such invocations.

The Batch Execution Mode

Entering the Batch Execution Mode, which actually invokes data for processing not 1 by 1 but in Batches of between 64 up to 912 rows

This might bring benefits in 10s & 100s times

Batch mode is essentially a Vector processing

Today’s Agenda I

Overall:
Not using the highest available compatibility level

Building:
Cutting on Memory
Using Strings
Partitioning without care
Compression Delay

Today’s Agenda II

Loading:

Not using Bulk Load API (<102.400 rows)

Identity

Merge

Parallel Loading

SSIS

Today's Agenda III

Querying

Casting on the wrong side

Overusing the NULL Expressions

Focusing on Cross Apply where it is not needed

Not using the Advanced Storage Engine Features

Estimated are similar! Similar How?

Trusting the triviality of the execution plans

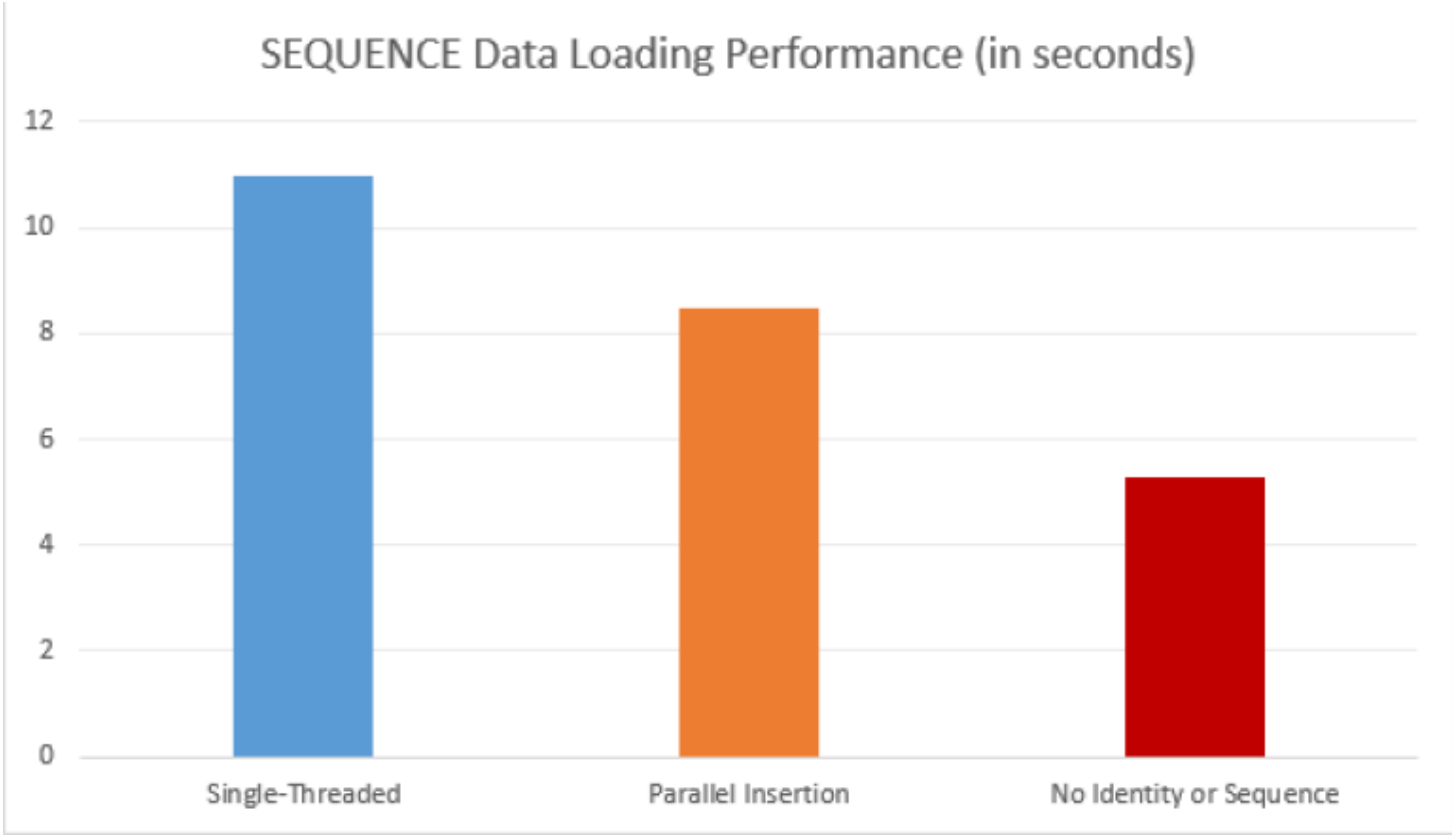
Today's Agenda IV

Using In-Memory CCI (Native Stored procs & tables for ETL insertion)

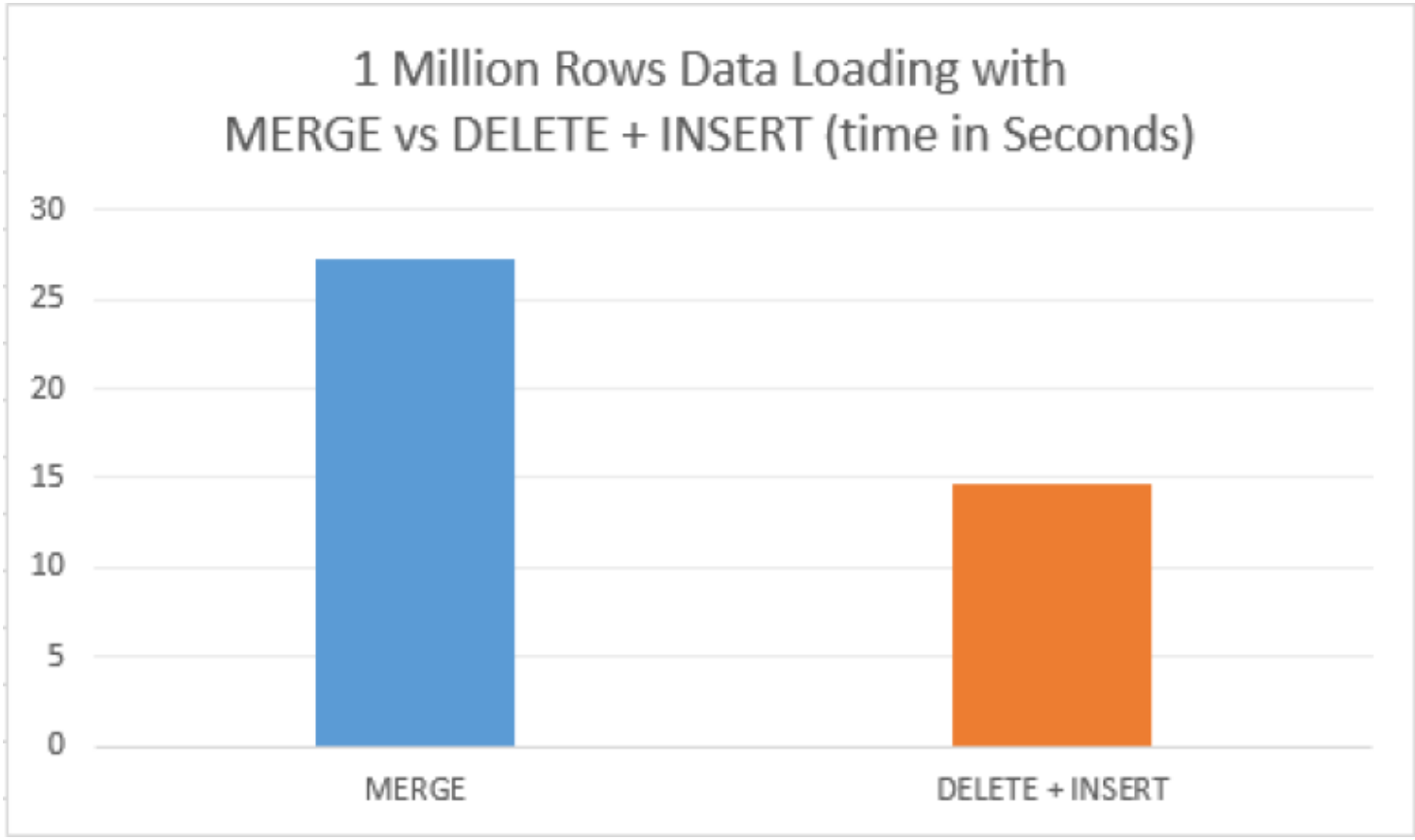
Relying too much on the automated Tuple Mover

Trusting ALTER INDEX … REORGANIZE completely

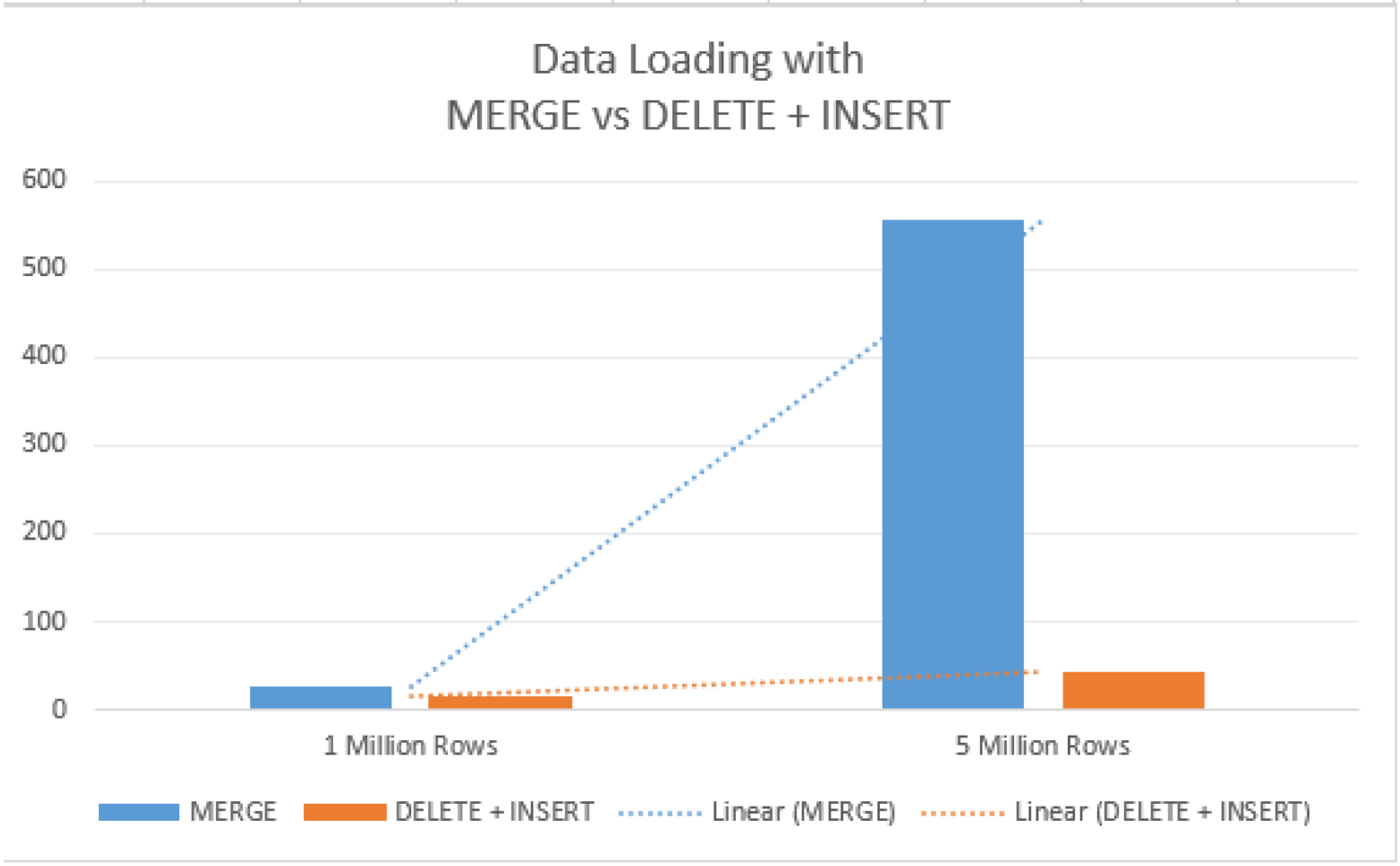
Identity


SEQUENCE Data Loading Performance (in seconds)
12
10
8
6
4
2
0
Single-Threaded
Parallel Insertion
No Identity or Sequence

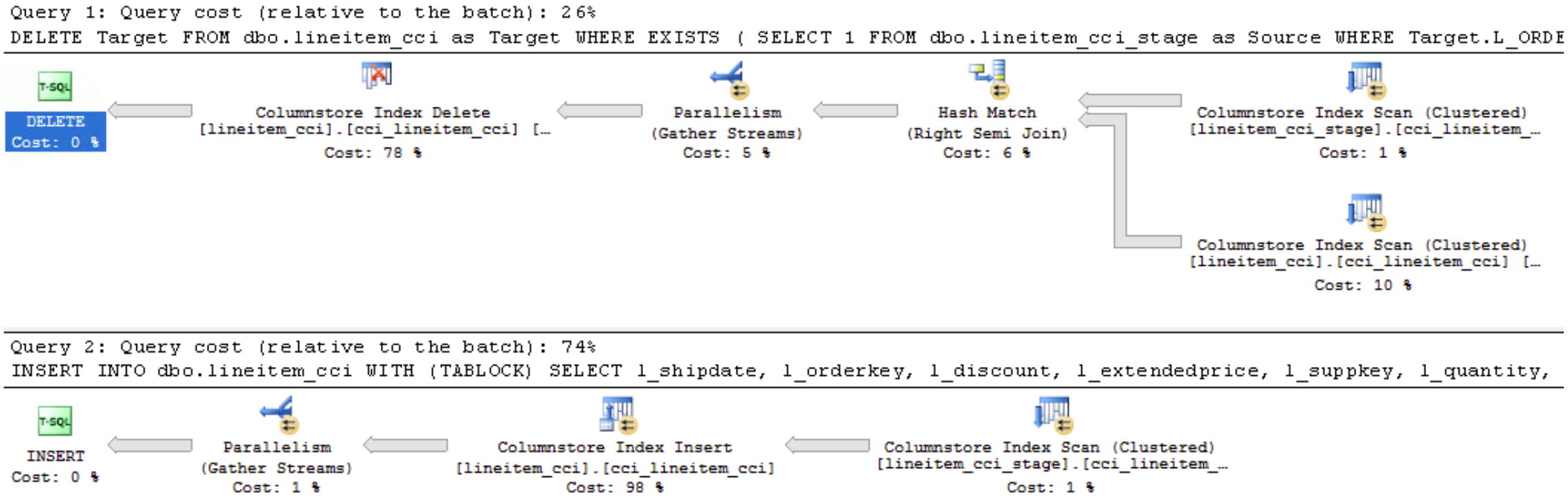
Merge


1 Million Rows Data Loading with
MERGE vs DELETE + INSERT (time in Seconds)
30
25
20
15
10
5
0
MERGE
DELETE + INSERT

Scaling Merge ?



Merge Execution Plans



Thank you very much!

Resources:

My Columnstore Blogpost Series (100+):

http://www.nikoport.com/columnstore

CISL – Open Source Columnstore Library:

https://github.com/NikoNeugebauer/CISL