

Fundamentals of Columnstore

Recap

Columnstore has 4 advantages:

- 1. Batch mode processing
- 2. Read less rows (row group & segment elimination)
- 3. Read less columns (especially from wide tables)
- 4. Compression

The more of these you can use, the better of a fit compression can be.



Stored in row groups & segments

Best visualized with sp_BlitzIndex:

l		partition_number	row_group_id	total_rows	deleted_rows	ld	AboutMe	Age	CreationDate	DisplayName	DownVotes .
ı	1	1	0	1048576	0	-1 to 1631413, 3 MB	4 to 173191, 0 MB	0 to 0.0 MB	170329813024768 to 176725041511585, 8 MB	4 to 779332, 3 MB	0 to 956712, 0 MB
ı	2	1	1	150621	0	1631414 to 1794633, 0 MB	4 to 17281, 0 MB	0 to 0, 0 MB	170407133214379 to 177008503420248, 1 MB	13 to 902148, 0	0 to 10830, 0 MB
l	3	1	2	983835	0	1794634 to 2847022, 3 MB	4 to 98108, 0 MB	0 to 0, 0 MB	170334123822289 to 178451611813389, 8 MB	4 to 865824, 3 MB	0 to 35834, 0 MB
l	4	1	3	997427	0	2847023 to 3917715, 3 MB	4 to 100565, 0 MB	0 to 0, 0 MB	170411434566073 to 179770161401029, 8 MB	4 to 913127, 3 MB	0 to 51237, 0 MB
ı	5	1	4	1015568	0	3917716 to 4993282, 3 MB	4 to 85464, 0 MB	0 to 0, 0 MB	170527397921729 to 181088711577554, 8 MB	4 to 844422, 3 MB	0 to 25915, 0 MB
l	6	1	5	1030248	0	4993283 to 6075397, 3 MB	4 to 73757, 0 MB	0 to 0, 0 MB	170535986852194 to 182295599153300, 8 MB	4 to 852335, 3 MB	0 to 16756, 0 MB
l	7	1	6	1010484	0	6075398 to 7127328, 3 MB	4 to 58227, 0 MB	0 to 0, 0 MB	172193842904823 to 183304926579629, 8 MB	4 to 851327, 3 MB	0 to 17546, 0 MB
l	8	1	7	1006297	0	7127329 to 8160389, 3 MB	4 to 50871, 0 MB	0 to 0, 0 MB	172279731490829 to 184245520282471, 8 MB	4 to 845259, 3 MB	0 to 3357, 0 MB
l	9	1	8	978221	0	8160390 to 9175559, 2 MB	4 to 43076, 0 MB	0 to 0, 0 MB	172155190811281 to 185125978054512, 7 MB	4 to 832592, 2 MB	0 to 6719, 0 MB
I	10	1	9	696230	0	9175560 to 9887118, 2 MB	4 to 27221, 0 MB	0 to 0.0 MB	171596846833396 to 185765931215757, 5 MB	4 to 605728, 2 MB	0 to 1133, 0 MB



Stored in row groups & segments

Best visualized with sp_BlitzIndex:

	and Ware and an		Antal arms	delega de		4.0		CreationDate	DisplayName	Down\/otes
	partition_number	row_group_id	total_rows	deleter		Ag	e	CreationDate	DisplayName	Downvotes
1	1	0	1048576		Row groups:	B 01	to 0, 0 MB	170329813024768 to 176725041511585, 8 MB	4 to 779332, 3 MB	0 to 956712, 0 MB
2	1	1	150621	0	max ~1M rows	0	to 0.0 MB	170407133214379 to 177008503420248, 1 MB	13 to 902148, 0	0 to 10830, 0 MB
3	1	2	983835	0	max ~1M rows	0	to 0, 0 MB	170334123822289 to 178451611813389, 8 MB	4 to 865824, 3 MB	0 to 35834, 0 MB
4	1	3	997427	0	1	B 0	to 0, 0 MB	170411434566073 to 179770161401029, 8 MB	4 to 913127, 3 MB	0 to 51237, 0 MB
5	1	4	1015568	0	331771010 4333262, 3 MB 410 63404, 0 MB	01	to 0, 0 MB	170527397921729 to 181088711577554, 8 MB	4 to 844422, 3 MB	0 to 25915, 0 MB
6	1	5	1030248	0	4993283 to 6075397, 3 MB 4 to 73757, 0 MB	01	to 0, 0 MB	170535986852194 to 182295599153300, 8 MB	4 to 852335, 3 MB	0 to 16756, 0 MB
7	1	6	1010484	0	6075398 to 7127328, 3 MB 4 to 58227, 0 MB	01	to 0, 0 MB	172193842904823 to 183304926579629, 8 MB	4 to 851327, 3 MB	0 to 17546, 0 MB
8	1	7	1006297	0	7127329 to 8160389, 3 MB 4 to 50871, 0 MB	01	to 0, 0 MB	172279731490829 to 184245520282471, 8 MB	4 to 845259, 3 MB	0 to 3357, 0 MB
9	1	8	978221	0	8160390 to 9175559, 2 MB 4 to 43076, 0 MB	01	to 0, 0 MB	172155190811281 to 185125978054512, 7 MB	41o 832592, 2 MB	0 to 6719, 0 MB
10	1	9	696230	0	9175560 to 9887118, 2 MB 4 to 27221, 0 MB	01	to 0.0 MB	171596846833396 to 185765931215757, 5 MB	4 to 605728, 2 MB	0 to 1133, 0 MB



Row groups may be "aligned"

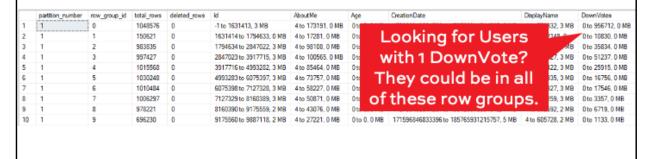
Based on how the source data was sorted when the columnstore index was initially created:





But other columns aren't distinct

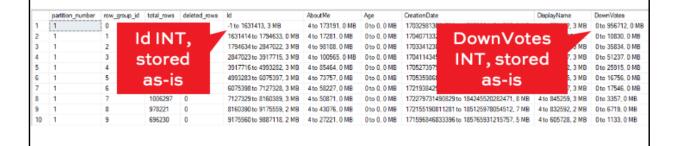
They're sorted, but think of it as 10 1M-row indexes each on the same column:





Integer columns are stored as-is

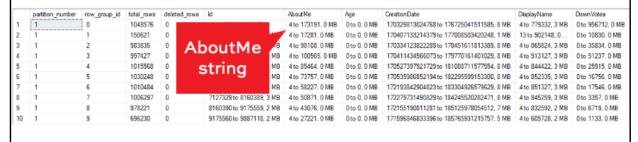
The integers can be stored directly in the indexes.





Other data types get dictionaries

The columnstore index holds integer values, and if you're looking for a specific value, you have to go look it up in the dictionary to find its ID:





As rows get deleted...

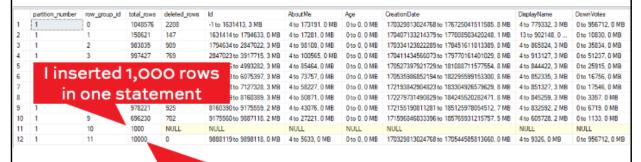
They still take up space in the columnstore indexes.

```
199 □ DELETE dbo.Users_columnstore
        200
                           WHERE DisplayName = 'alex';
        201
                      sp_BlitzIndex @TableName = 'Users_columnstore'
        202
⊞ Results 🖼 Messages
                                  total_rows
                                            deleted_rows
                                                                                   outMe
                                                                                                              Creation Date
                                                                                                                                                      DisplayName
                                                                                   to 173191, 0 MB
                                  1048576
                                            2208
                                                                                                 0 to 0, 0 MB 170329813024768 to 176725041511585, 8 MB 4 to 779332, 3 MB
                                                                                                                                                                      0 to 956712.0 MB
                                  150621
                                            147
                                                                                   to 17281, 0 MB
                                                                                                  0 to 0, 0 MB 170407133214379 to 177008503420248, 1 MB
                                                                                                                                                     13 to 902148. 0 ...
                                                                                                                                                                       0 to 10830, 0 MB
                                                              Extra
                                  983835
                                            909
                                                                                   to 98108, 0 MB
                                                                                                  0 to 0, 0 MB 170334123822289 to 178451611813389, 8 MB 4 to 865824, 3 MB
                                                                                                                                                                      0 to 35834, 0 MB
                                                                                  to 100565, 0 MB 0 to 0, 0 MB 170411434566073 to 179770161401029, 8 MB 4 to 913127, 3 MB 0 to 51237, 0 MB to 85464, 0 MB 0 to 0, 0 MB 170527397921729 to 181088711577554, 8 MB 4 to 844422, 3 MB 0 to 25915, 0 MB
                                  997427
                                             769
                                                         overhead
                                            1181
                                  1015568
                                  1030248
                                            1145
                                                                                   to 73757, 0 MB
                                                                                                  0 to 0, 0 MB 170535986852194 to 182295599153300, 8 MB 4 to 852335, 3 MB
                                                                                                                                                                      0 to 16756, 0 MB
                                                                                   to 58227, 0 MB
                                                                                                  0 to 0, 0 MB 172193842904823 to 183304926579629, 8 MB 4 to 851327, 3 MB 0 to 17546, 0 MB
                                  1010484
                                            941
                                  1006297
                                            1006
                                                         7127329 to 8160389, 3 MB 4 to 50871, 0 MB
                                                                                                  0 to 0, 0 MB 172279731490829 to 184245520282471, 8 MB 4 to 845259, 3 MB
                                                                                                                                                                      0 to 3357, 0 MB
                                                         8160390 to 9175559, 2 MB 4 to 43076, 0 MB
                                                                                                  Oto 0, 0 MB 172155190811281 to 185125978054512, 7 MB 4 to 832592, 2 MB Oto 6719, 0 MB
                                  978221
                                                         9175560 to 9887118, 2 MB 4 to 27221, 0 MB
                                                                                                  O to 0, 0 MB 171596846833396 to 185765931215757, 5 MB 4 to 605728, 2 MB 0 to 1133, 0 MB
```

As rows get inserted...

In small quantities: they go into a heap (delta store)

In larger quantities: go into their own columnstores



Then 10,000 rows in another statement

Updates just plain ol' suck

They work like delete + insert:

- · The rows are deleted in their original row group
- · Then inserted as a new rows

But they don't perform as well as delete + insert.

Microsoft may change that in the future, but right now, updates are a bad idea in columnstore.



Selects

In small quantities: they go into a heap (delta store)

In larger quantities: go into their own columnstores

	partition_number	row_group_id	total_rows	deleted_rows	ld	AboutMe	Age	CreationDate	DisplayName	DownVotes
1	1	0	1048576	2208	-1 to 1631413, 3 MB	4 to 173191, 0 MB	0 to 0, 0 MB	170329813024768 to 176725041511585, 8 MB	4 to 779332, 3 MB	0 to 956712, 0 MB
2	1	1	150621	147	1631414 to 1794633, 0 MB	4 to 17281, 0 MB	0 to 0, 0 MB	170407133214379 to 177008503420248, 1 MB	13 to 902148, 0	0 to 10830, 0 MB
3	1	2	983835	909	1794634 to 2847022, 3 MB	4 to 98108, 0 MB	0 to 0, 0 MB	170334123822289 to 178451611813389, 8 MB	4 to 865824, 3 MB	0 to 35834, 0 MB
4	1	3	997427	769	2847023 to 3917715, 3 MB	4 to 100565, 0 MB	0 to 0, 0 MB	170411434566073 to 179770161401029, 8 MB	4 to 913127, 3 MB	0 to 51237, 0 MB
5	Line	المصاسي	4 0 0		to 4993282, 3 MB	4 to 85464, 0 MB	0 to 0, 0 MB	170527397921729 to 181088711577554, 8 MB	4 to 844422, 3 MB	0 to 25915, 0 MB
6	⊓ins€	ertea	1,00	O rov	3 to 6075397, 3 MB	4 to 73757, 0 MB	0 to 0, 0 MB	170535986852194 to 182295599153300, 8 MB	4 to 852335, 3 MB	0 to 16756, 0 MB
7	t in a				8 to 7127328, 3 MB	4 to 58227, 0 MB	0 to 0, 0 MB	172193842904823 to 183304926579629, 8 MB	4 to 851327, 3 MB	0 to 17546, 0 MB
8	in (one s	tate	ment	9 to 8160389, 3 MB	4 to 50871, 0 MB	0 to 0.0 MB	172279731490829 to 184245520282471, 8 MB	4 to 845259, 3 MB	0 to 3357, 0 MB
9	1		978221	925	8160390 to 9175559, 2 MB	4 to 43076, 0 MB	0 to 0.0 MB	172155190811281 to 185125978054512, 7 MB	4 to 832592, 2 MB	0 to 6719, 0 MB
10	1	9	696230	702	9175560 to 9887118, 2 MB	4 to 27221, 0 MB	0 to 0, 0 MB	171596846833396 to 185765931215757, 5 MB	4 to 605728, 2 MB	0 to 1133, 0 MB
11	1	10	1000	NULL	NULL	NULL	NULL	NULL	NULL	NULL
12	1	11	10000	0	9888119 to 9898118, 0 MB	4 to 5633, 0 MB	0 to 0, 0 MB	170329813024768 to 170544585813660, 0 MB	4 to 9326, 0 MB	0 to 956712, 0 MB

Then 10,000 rows in another statement

This is why columnstore degrades.

You do a proof-of-concept by building a new columnstore index once.

The data's packed nicely into big row groups.

Performance is super fast when you demo.

Over time, as you insert/update/delete data, you have a ton of small, non-aligned row groups, and select performance degrades.



Index maintenance is a big deal.

Index maintenance can get you:

- Fewer row groups (especially important if you're inserting/modifying/deleting <1M rows at a time)
- · Segment elimination for one column (if you plan)

But it's way harder because:

- · Rebuilds are much more CPU-intensive
- Multi-threaded rebuilds aren't aligned
- The source data needs to be sorted on the column where you want segment elimination



Ways to work around it

If you just want batch mode, try an empty filtered nonclustered columnstore index.

If you're on 2019, go into 2019 compat level for batch mode execution on rowstore.

If you only need analytics on a subset of columns, and their contents don't change frequently, try a nonclustered columnstore index.

But if you have ~1B rows or 100GB+ data...



Partitioning helps big data, big time

Gets you partition elimination if you filter by a date

Gets you way, way easier index maintenance by letting you export/sort/rebuild 1 partition at a time

But in 2020, there aren't community tools yet to make this process easier.



Don't let your implementation fail

Start with the quiz: https://ColumnScore.com

Make a list of the queries that need to perform

Design which column will need segment elimination

Do the initial table load with sorted data

Run your typical insert/update/delete workloads

Do your index maintenance

Test selects AFTER the loads & maintenance too



Next steps for learning

Niko Neugebauer: http://www.nikoport.com/columnstore/

Joe Obbish:

https://www.erikdarlingdata.com/author/joe-obbish/

My bookmarks:

https://pinboard.in/u:brento/t:columnstore/

The documentation, seriously:

https://docs.microsoft.com/en-us/sql/relationaldatabases/indexes/columnstore-indexes-overview



You can do this.

For questions, leave comments on the relevant module.

For private help after the class, email Help@BrentOzar.com with:

- · A note that you were in this class
- sp_Blitz @CheckServerInfo = 1
- sp_BlitzFirst @SinceStartup= 1



