Row-based vs. Column-based Indexes

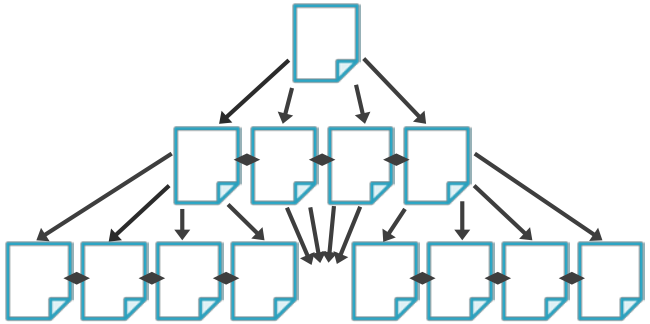
Kimberly L. Tripp
OWNER/PRESIDENT – SQLSKILLS.COM
@kimberlyltripp www.sqlskills.com/blogs/kimberly

Module Overview

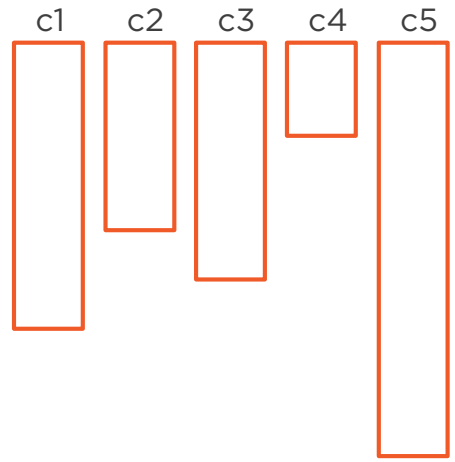
- Row-based index structures
- Column-based index structures
- Understanding your workload
- Row-based vs. column-based indexes
- Columnstore indexes by version
- Indexed views vs. columnstore indexes

Table Structures

Heap structure
(possible in all versions)

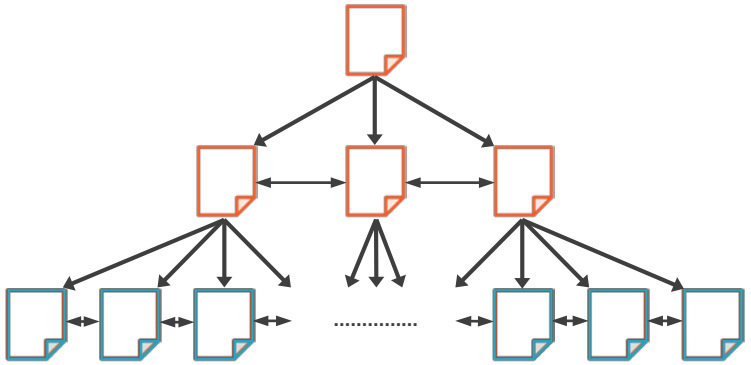


Row-based clustered index
(possible in all versions)



Column-based clustered index
(limited options in SQL Server 2014; more options in SQL Server 2016+)

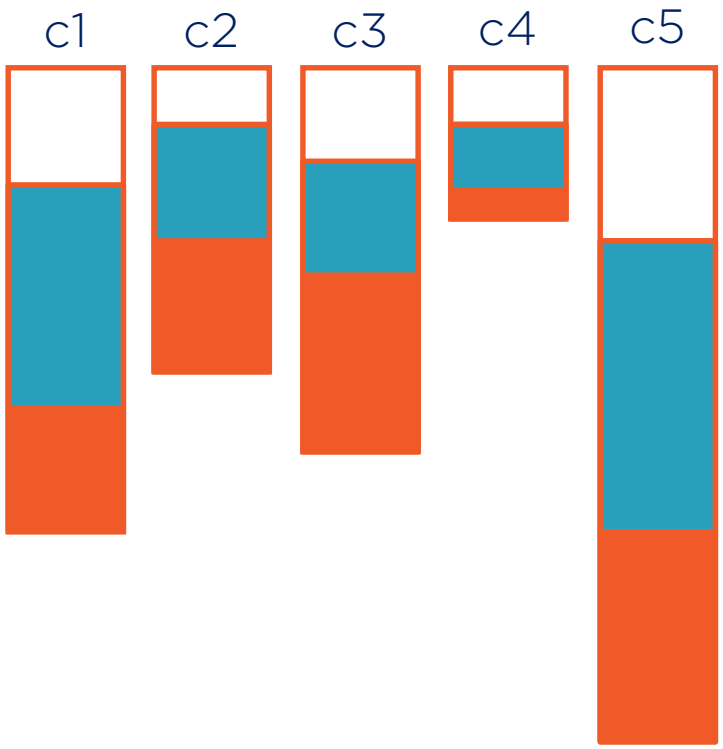
Index Structures: Row-based Indexes



Leaf (bottom) level: contains something for every table row in indexed order

Non-leaf level(s) or B-tree: contains something, specifically representing the FIRST value, from every page of the level below

Index Structures: Column-based Indexes



All rows for a single column are stored together – potentially HIGHLY compressible!

Data is segmented into groups of 1 million rows for better batch processing

SQL Server can do “segment elimination” (similar to partition elimination) and further reduce number of segments to process!

Parallelization through batch mode

Choice Depends on Workload: OLTP

Priority is toward modifications

Many point queries (highly-selective and return very few rows)

Row-based clustered index often best

Row-based nonclustered indexes are very important

Might use nonclustered columnstore indexes

Choice Depends on Workload: DSS/RDW

Priority is toward large-scale aggregates

High percentage of or entire dataset is evaluated often

Clustered columnstore for aggregates and whole-table queries

Secondary indexes still important

Point queries benefit from row-based nonclustered indexes

Choice Depends on Workload: Hybrid

OLTP might be priority, with some point query activity

Some range-based queries because "management" wants real-time analysis

Needs mix of indexes across tables

Possibly nonclustered columnstore indexes if data is partitioned

Best in SQL Server 2016+

DSS/RDW Strategies by Version

Prior to SQL Server 2012

- Traditional row-based clustered and nonclustered indexes

SQL Server 2012+

- Consider adding a read-only, nonclustered, columnstore index for partitioned objects leveraging partition switching as additional data is added

SQL Server 2014+

- Use SQL Server 2012+ strategy above
- Or consider the new updateable clustered columnstore index

RDW/Columnstore Indexes by Version

- **SQL Server 2008**
  - Lowest version to consider for large table/data analysis with performance and scalability features
    - Added data compression (row and page compression)
    - Added filtered indexes/filtered statistics
    - Fixed fast-switching for partition-aligned, indexed views
- **SQL Server 2012**
  - Added read-only, nonclustered columnstore indexes
  - Some frustrating limitations but still amazing performance when possible and/or workarounds used (details at http://bit.ly/1eHVW00)

RDW/Columnstore Indexes by Version

- SQL Server 2014
  - Added updateable, clustered columnstore indexes
    - Frustrating limitation that no other indexes allowed
  - Many other frustrating limitations with columnstore fixed
    - E.g., UNION ALL supports batch mode, so useable with partitioned views
  - Added “incremental statistics” to help limit rebuilds as well as time to rebuild
- SQL Server 2016
  - Added updateable nonclustered columnstore indexes
  - Added row-based nonclustered indexes with clustered columnstore indexes
  - Any combination of base-table structure with nonclustered indexes possible

Row-based vs. Column-based Indexes

Row-based	Column-based
Support data compression	Columnar data stored together, often allows much higher level of compression

Row-based vs. Column-based Indexes

Row-based	Column-based
Support data compression	Columnar data stored together, often allows much higher level of compression
Can support point queries (seeks)	Supports large-scale aggregations

Row-based vs. Column-based Indexes

Row-based	Column-based
Support data compression	Columnar data stored together, often allows much higher level of compression
Can support point queries (seeks)	Supports large-scale aggregations
Wide variety of supported scans	Support partial scans with segment elimination, and combine with partitioning for further elimination
Full and/or partial table scans	
Nonclustered covering scans	
Nonclustered covering seeks with partial scans	

Row-based vs. Column-based Indexes

Row-based Problems

More tuning work for analysis: must create appropriate indexes per query and then consolidate

Stores multiple columns of data together (not as easily compressed)

Column-based Problems

Minimum set for reads is a row group (no seeks possible)

Limitations of features for batch mode by version (fixes in 2014 and 2016)

Limitations with other features (fewer and fewer by SQL Server version)

Indexed Views vs. Columnstore Indexes

Indexed Views

Limited uses in non-Enterprise Editions

Columnstore Indexes

Enterprise Edition only before 2016 SP1

Indexed Views vs. Columnstore Indexes

Indexed Views	Columnstore Indexes
Limited uses in non-Enterprise Editions	Enterprise Edition only before 2016 SP1
Requires certain session settings set on	No session setting requirements

Indexed Views vs. Columnstore Indexes

Indexed Views	Columnstore Indexes
Limited uses in non-Enterprise Editions	Enterprise Edition only before 2016 SP1
Requires certain session settings set on	No session setting requirements
Must be analyzed / created “per query”	**Only one can be created per table**

Indexed Views vs. Columnstore Indexes

Indexed Views	Columnstore Indexes
Limited uses in non-Enterprise Editions	Enterprise Edition only before 2016 SP1
Requires certain session settings set on	No session setting requirements
Must be analyzed / created “per query”	Only one can be created per table
More complicated to create	**Very easy to create**

Indexed Views vs. Columnstore Indexes

Indexed Views	Columnstore Indexes
Limited uses in non-Enterprise Editions	Enterprise Edition only before 2016 SP1
Requires certain session settings set on	No session setting requirements
Must be analyzed / created “per query”	Only one can be created per table
More complicated to create	Very easy to create
More storage required	**A LOT LESS storage required**

Indexed Views vs. Columnstore Indexes

Indexed Views	Columnstore Indexes
Limited uses in non-Enterprise Editions	Enterprise Edition only before 2016 SP1
Requires certain session settings set on	No session setting requirements
Must be analyzed / created “per query”	Only one can be created per table
More complicated to create	Very easy to create
More storage required	A LOT LESS storage required
More administrative overhead / maintenance	**Less administrative overhead / maintenance**

Indexed Views vs. Columnstore Indexes

Indexed Views	Columnstore Indexes
Limited uses in non-Enterprise Editions	Enterprise Edition only before 2016 SP1
Requires certain session settings set on	No session setting requirements
Must be analyzed / created “per query”	Only one can be created per table
More complicated to create	Very easy to create
More storage required	A LOT LESS storage required
More administrative overhead / maintenance	Less administrative overhead / maintenance
More costly to maintain during inserts / updates	**Might not be able to do inserts / updates, depending on version**

Summary

Understanding ALL index types is really best as you move forward

This course will focus on row-based indexes

For columnstore indexes:

- Pluralsight: *SQL Server 2012: Nonclustered Columnstore Indexes*
- Blog series: http://www.nikoport.com/columnstore/

What We Covered

- Row-based index structures
- Column-based index structures
- Understanding your workload
- Row-based vs. column-based indexes
- Columnstore indexes by version
- Indexed views vs. columnstore indexes