Clustered Columnstore Indexes

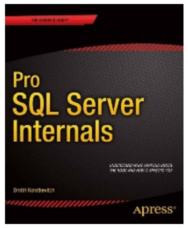
Internals and Design Considerations

About me

- 20+ years of experience in IT
- 14+ years of experience working with SQL Server

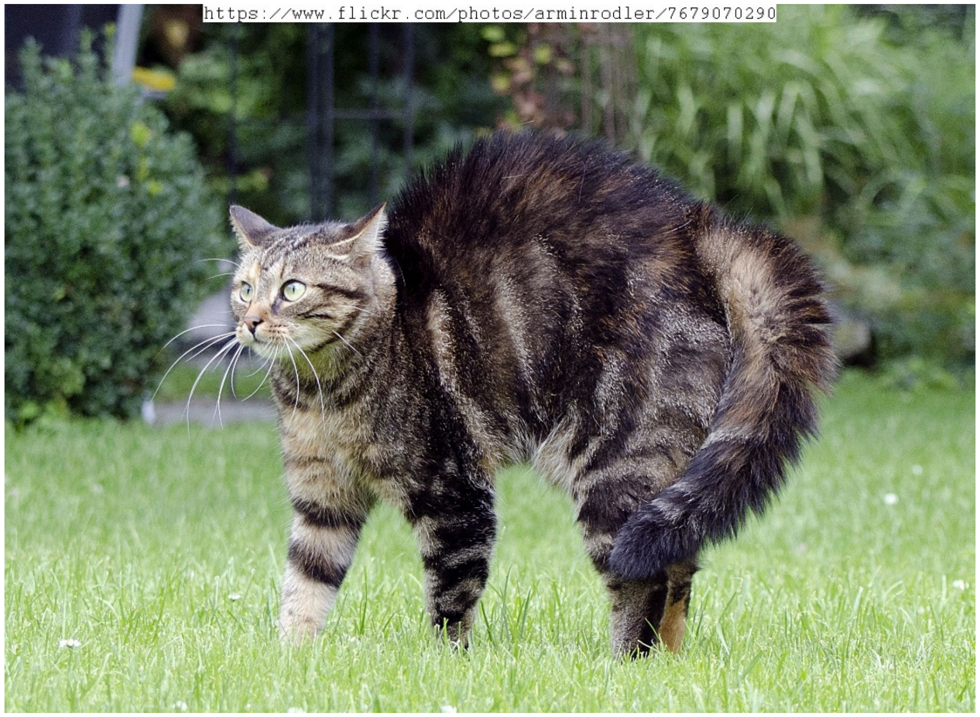
- Microsoft SQL Server MVP
- Microsoft Certified Master (SQL Server 2008)

- Blog: http://aboutsqlserver.com
- Email: dmitri@aboutsqlserver.com



Serious BI folks

Frightened OLTP guy



Agenda

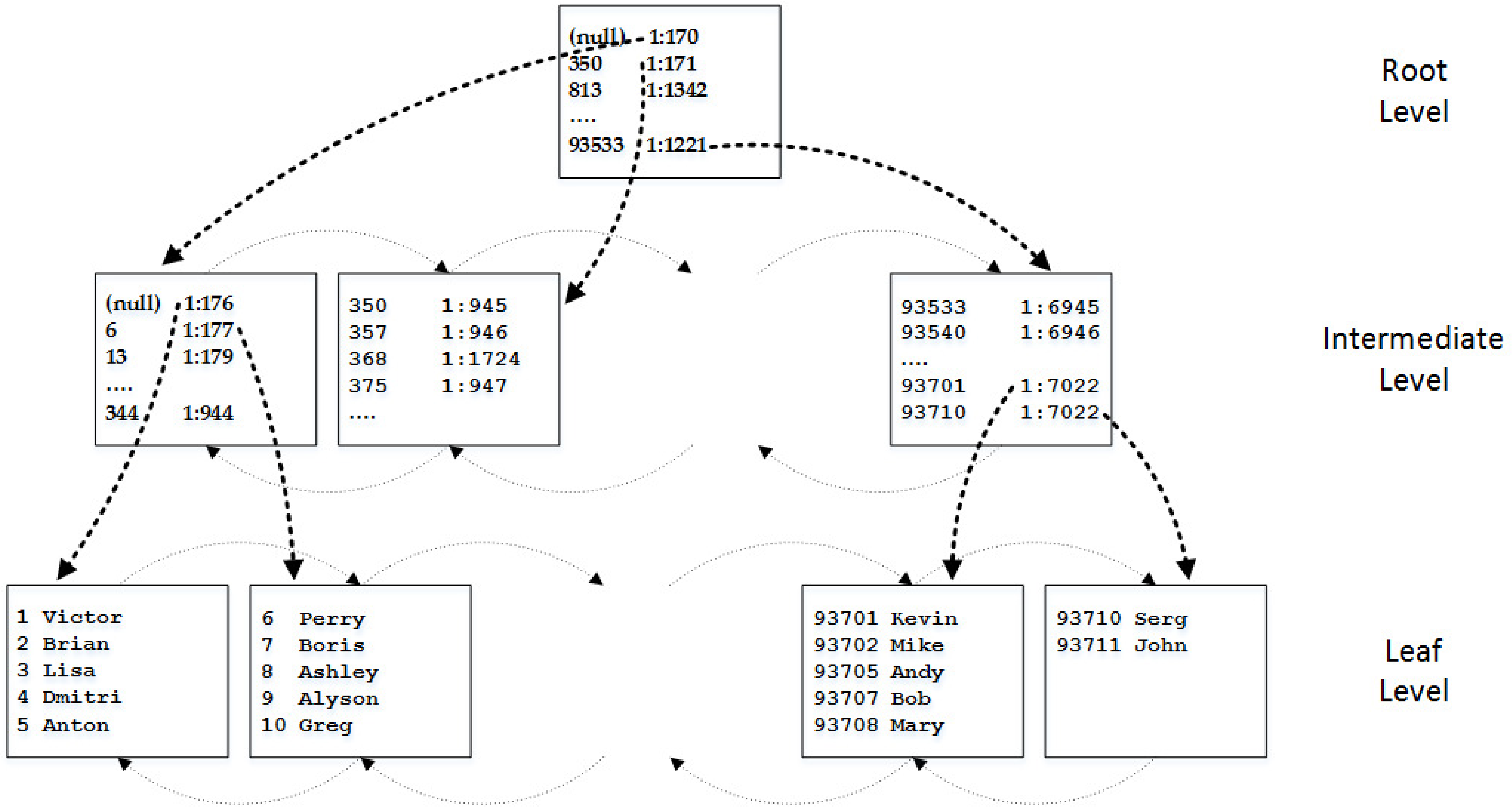
In Scope

- Clustered Columnstore Indexes Internals
- Performance gems and performance hogs
- Maintenance and Design considerations

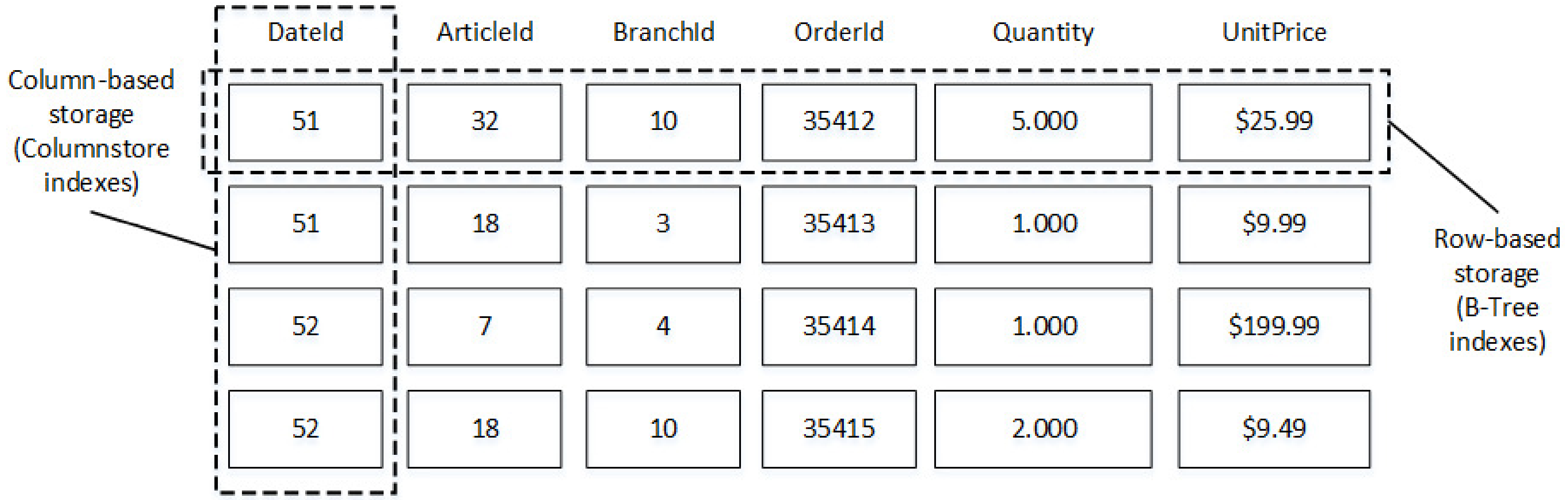
Out of Scope

- Data Warehouse design considerations
- Batch-mode execution

B-Tree Indexes



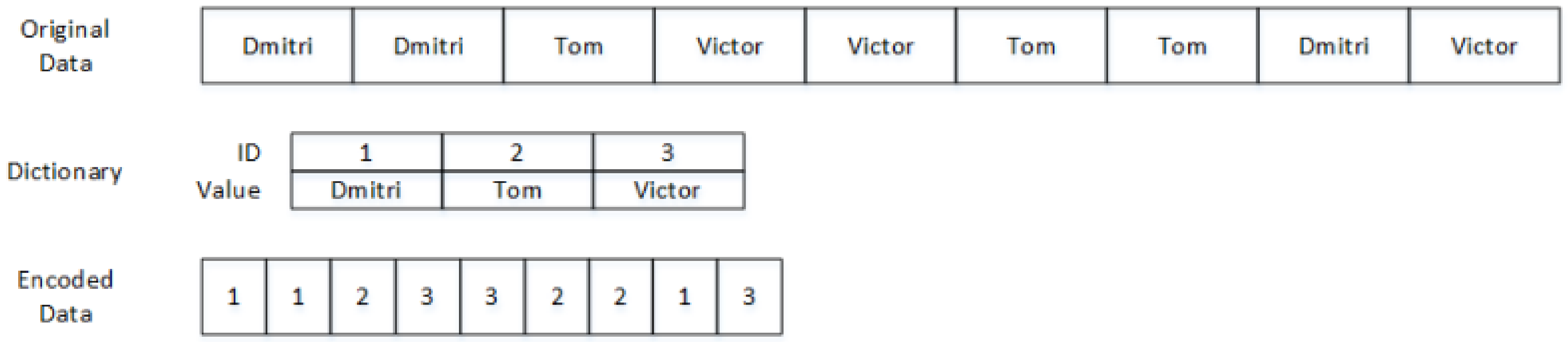
Row-Based and Column-Based Storage



DateId	ArticleId	BranchId	OrderId	Quantity	UnitPrice
51	32	10	35412	5.000	$25.99
51	18	3	35413	1.000	$9.99
52	7	4	35414	1.000	$199.99
52	18	10	35415	2.000	$9.49

Column-based storage (Columnstore indexes)
Row-based storage (B-Tree indexes)

Columnstore Compression (Dictionary Encoding)

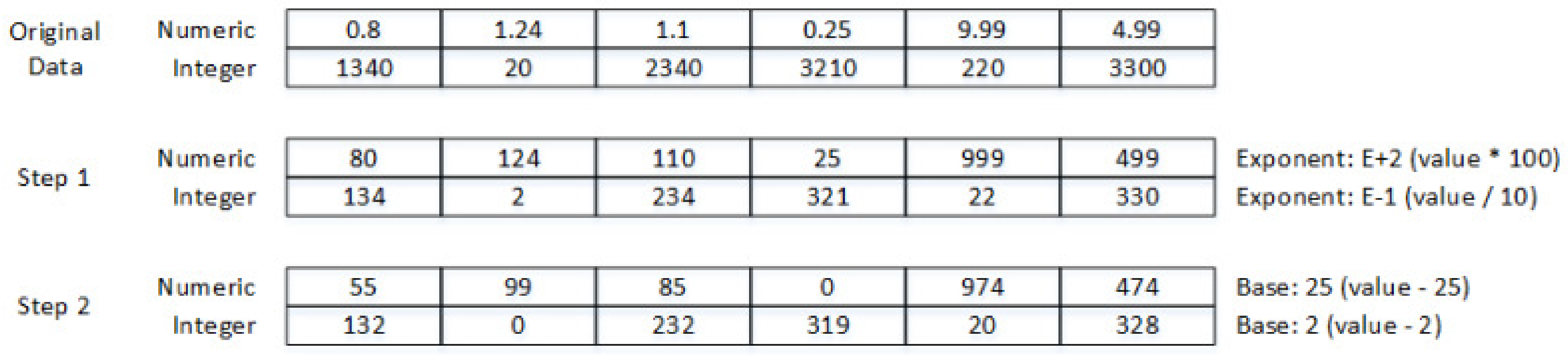


Original Data									
	Dmitri	Dmitri	Tom	Victor	Victor	Tom	Tom	Dmitri	Victor

Dictionary				
	ID	1	2	3
	Value	Dmitri	Tom	Victor

Encoded Data									
	1	1	2	3	3	2	2	1	3

Columnstore Compression (Value-Based Encoding)



Original Data	Numeric	0.8	1.24	1.1	0.25	9.99	4.99	
	Integer	1340	20	2340	3210	220	3300	
Step 1	Numeric	80	124	110	25	999	499	Exponent: E+2 (value * 100)
	Integer	134	2	234	321	22	330	Exponent: E-1 (value / 10)
Step 2	Numeric	55	99	85	0	974	474	Base: 25 (value - 25)
	Integer	132	0	232	319	20	328	Base: 2 (value - 2)

Demo

COMPRESSION AND DATA SIZE

Compression in SQL Server

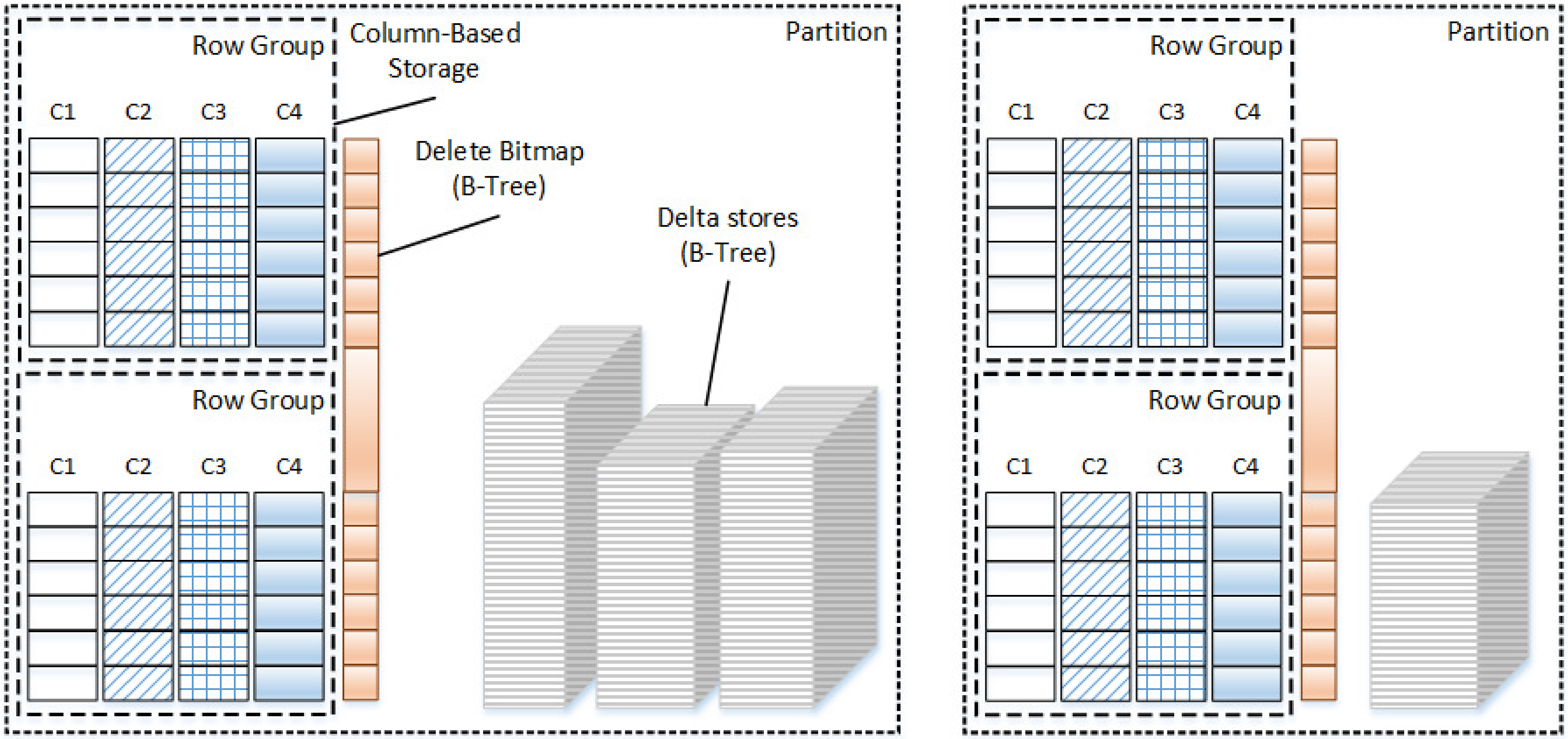
Type	Demo Data Size (MB)*	Description
No Compression	1,633MB	Fixed-Length types use the same space even when NULL
ROW	862MB	Fixed-Length types storage space varies based on the data. Introduces slight CPU overhead, which is offset by I/O improvement
PAGE	378MB	ROW + Prefix + Dictionary compression on single data page scope. Good for static data
Columnstore	123MB	Dictionary + Value-Based encoding on row-group level
Columnstore_Archive	31MB	Columnstore + ZIP

(*) Actual compression results would vary based on the schema, indexes and data

Demo

ARCHIVE COMPRESSION

Clustered Columnstore Indexes



- Delete Bitmap indicates what rows were deleted
- Delta Store stores inserted and updated rows (Max 1,048,576 rows)

Demo

CCI MODIFICATION INTERNALS

Data Load Performance

- “Trickle” (regular) inserts go to delta store
- Bulk inserts (bulk API) go to:
  - row groups if batch size > ~100,000
  - delta store otherwise

Batch Size	Row Groups	Delta Store
99,999	0	99,999
150,000	150,000	0
1,048,577	1,048,576	1
2,100,000	1,048,576; 1,048,576	2,848
2,250,000	1,048,576; 1,048,576; 152,848	0

Demo

BATCH SIZE AND INSERT PERFORMANCE

Factors Affecting Performance

- Large number of small row groups
- Large Delta Stores
- Large Delete Bitmaps

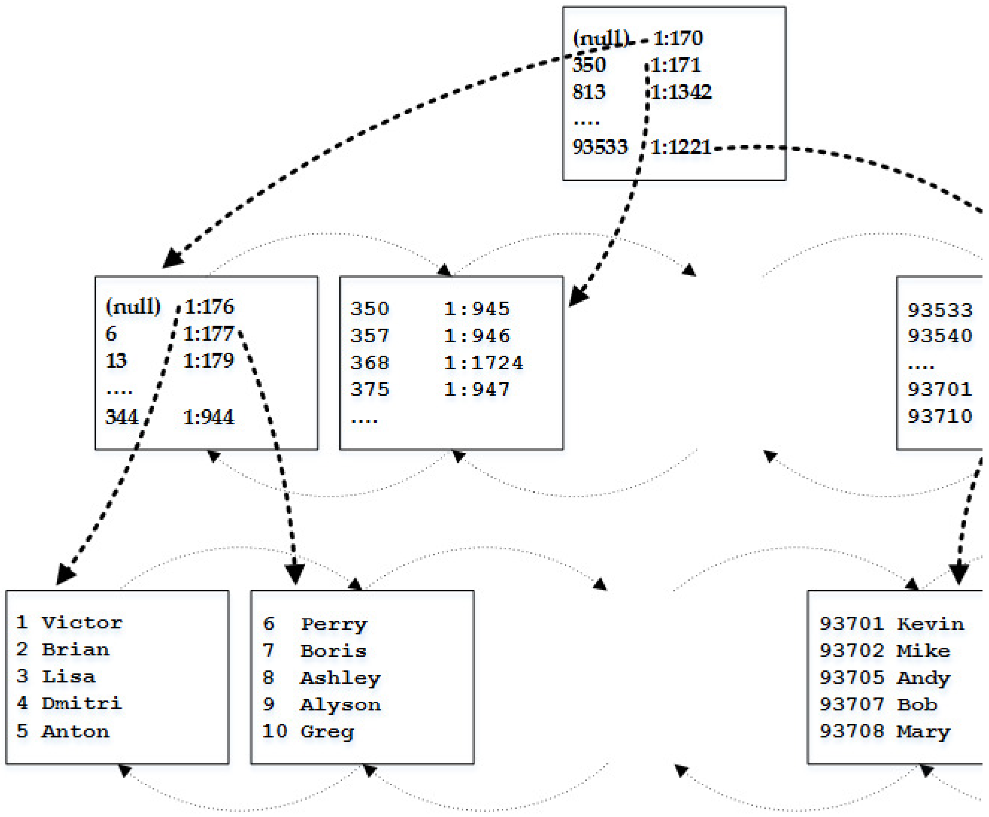
Demo

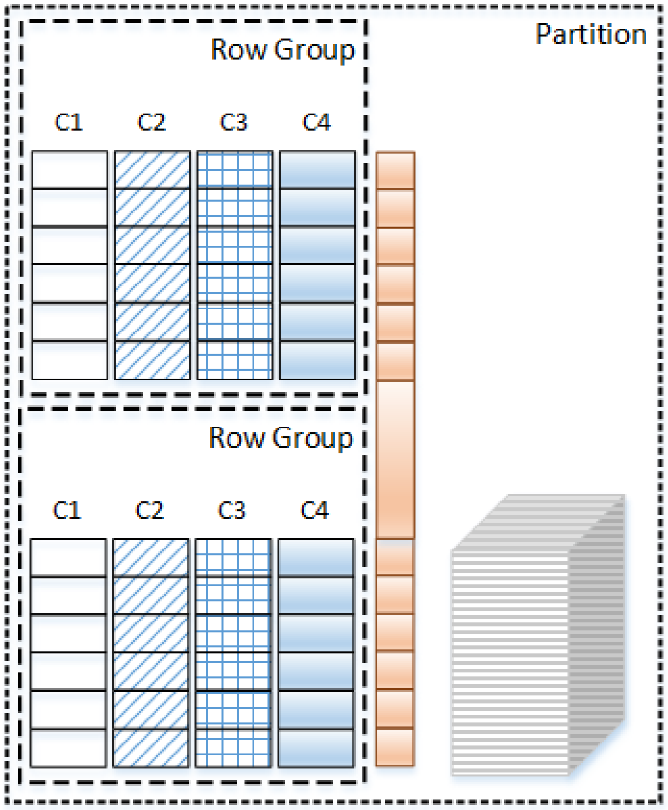
FACTORS THAT ARE AFFECTING PERFORMANCE

Index Maintenance

- ALTER INDEX REORGANIZE converts closed delta stores to the row groups
  - T*uple mover* process running on-demand
  - Data can be inserted but not modified/deleted
- ALTER INDEX REBUILD rebuilds the index
  - Removes deleted rows
  - Merges row groups and delta stores
  - Data can be read but cannot be modified
- Maintenance can be done on per-partition basis

Access Patterns





Using CCI

Good for..

- Reporting and Analysis queries that scan large amount of data
- ETL process that insert large amount of data
- Fact and large Dimension tables in DW

Bad for..

- Small range scans and singleton lookups
- Data modifications in the small batches
- Transactional and Catalog entities in OLTP

Data Partitioning

- Use table partitioning with CCI
  - Limit # of partitions affected by ETL processes
  - Rebuild affected partitions after ETL
- In OLTP systems consider to use partition views
  - Volatile active data in B-Tree indexes
  - Old static data in columnstore indexes
    - Nonclustered columnstore indexes could be the better option in the mixed workload

Key Points

- Use CCI only for the valid use-cases
- Import data to CCI using bulk API with the batches close to 1,048,576 rows
- Keep delta stores and delete bitmap as small as possible
- Partition the tables with CCI and rebuild affected partitions after ETL processes
- Combine CI and B-Tree indexes with partitioned views in OLTP (when appropriate)

Q & A

- Thank you very much for attending!

- Slides and scripts are available: http://aboutsqlserver.com/presentations

- Email: dmitri@aboutsqlserver.com