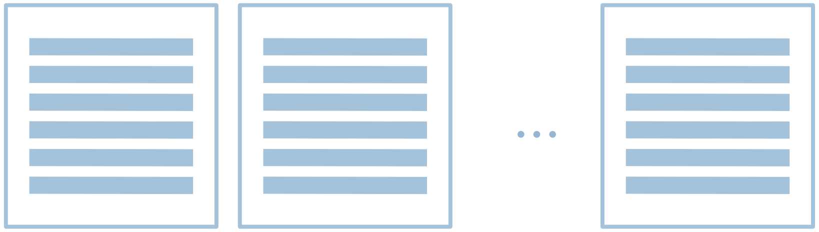
# ENHANCEMENTS TO SQL SERVER COLUMN STORE INDEXES

**Paul Larson**, Cipri Clinciu, Campbell Fraser, Eric N. Hanson,
Mostafa Mokhtar, Michal Nowakiewicz, Vassilis Papadimos,
Susan L. Price, Srikumar Rangarajan, Remus Rusanu, Mayukh Saubhasik
Microsoft

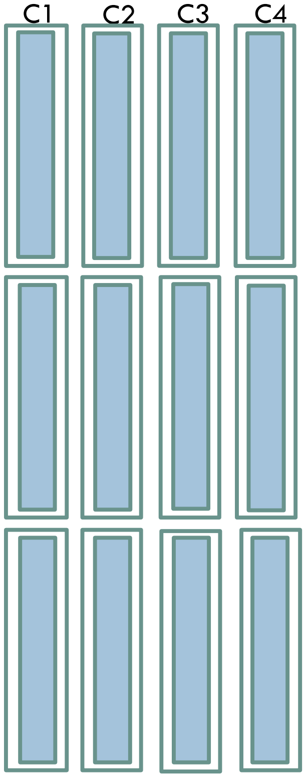
# Outline

- Column store indexes in SQL Server 2012
- Updatable clustered column store
- Query processing enhancements
- Archival compression
- Performance examples
- Status

# What is a column store index?



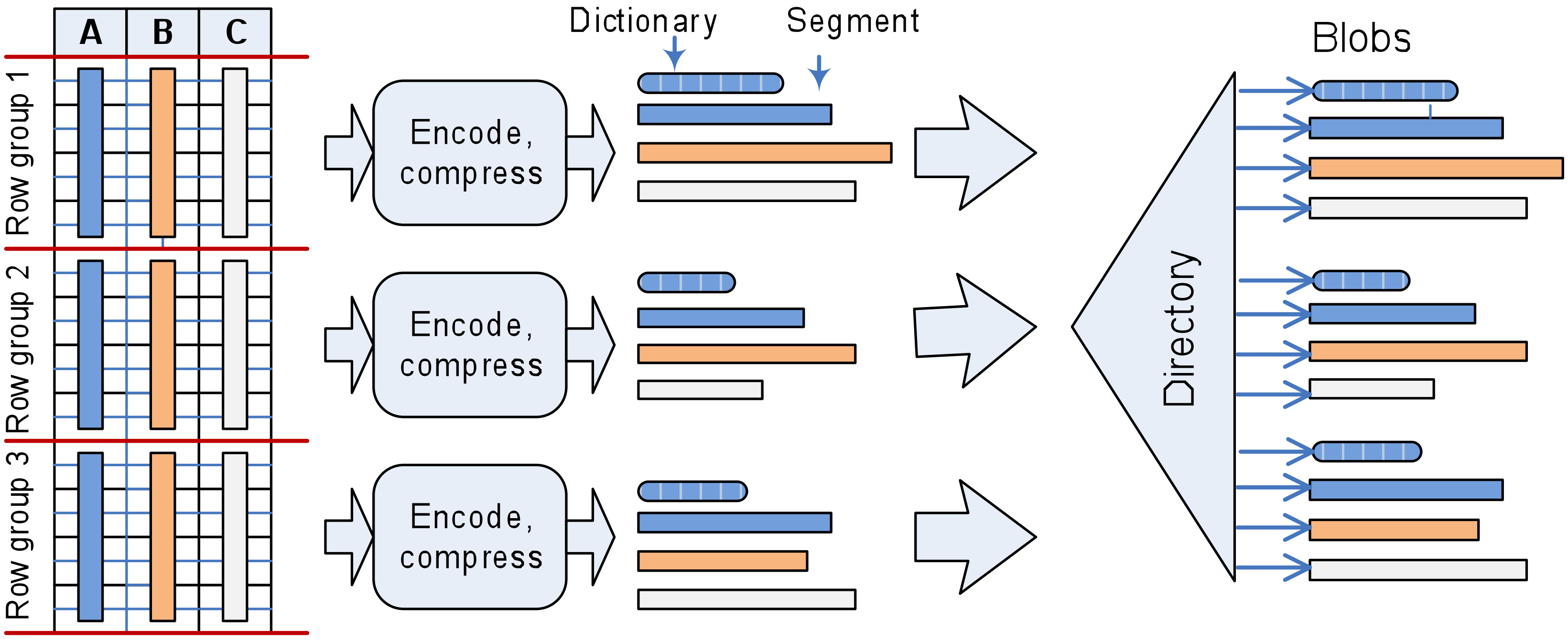
A B-tree index stores data row-wise



A column store index stores data column-wise

- Each page stores data from a single column
- Data <u>not</u> stored in sorted order
- Optimized for scans

# Index creation and storage



- □ Also have a global dictionary per column (not shown)
- □ Degree of parallelism dynamically adjusted based on memory availability

# Column store compression

- Encoding – convert to integers
  - Value-based encoding
  - Dictionary (hash) encoding
- Row reordering
  - Find optimal ordering of rows (best compression)
  - Proprietary algorithm (VertiPaq)
- Compression
  - Run length encoding (value + number of consecutive repeats)
  - Bit packing (use min number of bits)

# Observed compression ratios

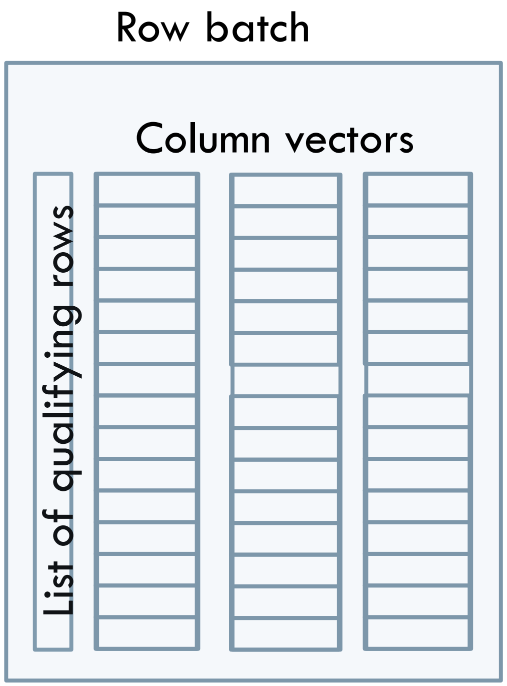
| Data Set | Uncompressed table size (MB) | Column store index size (MB) | Compression Ratio |
|---|---|---|---|
| Cosmetics | 1,302 | 88.5 | 14.7 |
| SQM | 1,431 | 166 | 8.6 |
| Xbox | 1,045 | 202 | 5.2 |
| MSSales | 642,000 | 126,000 | 5.1 |
| Web Analytics | 2,560 | 553 | 4.6 |
| Telecom | 2,905 | 727 | 4.0 |

1.8X better compression than SQL's page compression

# IO and caching

- New (large) object cache
  - Cache for column segments and dictionaries
- Aggressive read ahead
  - At segment level
  - At page level within a segment
- Early segment elimination based on segment metadata
  - Min and max values stored in segment metadata

# Batch mode processing



- Process a batch of rows at a time
  - Batch stored in vector form
- Batch mode operators in SQL 2012
  - Filter, (inner) hash join, (local)hash aggregation
- Greatly reduced CPU time

# Example query

```
select w_city, w_state, d_year,
       SUM(cs_sales_price) as cs_sales_price
from warehouse, catalog_sales, date_dim
where w_warehouse_sk = cs_warehouse_sk
  and cs_sold_date_sk = d_date_sk
  and w_state = 'SD' and d_year = 2002
group by w_city, w_state, d_year
order by d_year, w_state, w_city;
```

**1TB TPC-DS database**

| | |
|---|---|
| Catalog_Sales | 1.44B rows |
| Warehouse | 20 rows |
| Date_dim | 73,049 rows |

**Machine:** 40/80 cores/threads, 256 GB, IO bandwidth 10GB/sec

| | Cold buffer pool | | Warm buffer pool | |
|---|---|---|---|---|
| | CPU | Elapsed | CPU | Elapsed |
| **Row store only** | 259 | 20 | 206 | 3.1 |
| **Column store** | 19.8 | 0.8 | 16.3 | 0.3 |
| **Improvement** | 13X | 25X | 13X | 10X |

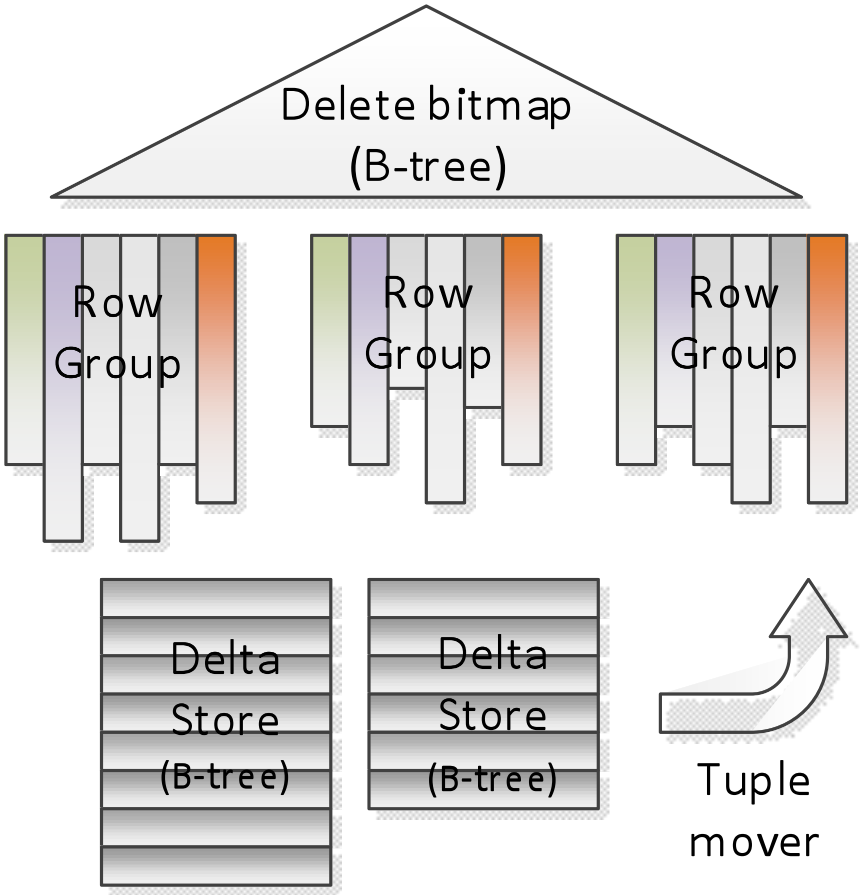
# Outline

- Column store indexes in SQL Server 2012
- Updatable clustered column store
- Query processing enhancements
- Archival compression
- Performance examples
- Status

# Clustered column store index

- Secondary index only in SQL 2012
- Can now be used as primary store for a table
  - Clustered index in SQL Server parlance
  - Very significant storage savings
- Fully updatable
- Sampling and statistics support
  - Two-level sampling (row groups + rows)
  - True random row sampling
    - Used for computing stats – much better accuracy

# Update mechanisms



- □ Delete bitmap
  - B-tree on disk
  - Bitmap in memory
- □ Delta stores
  - Up to 1M rows/store
  - May have several
- □ Tuple mover
  - Converts delta store to row group
  - Automatically or on demand

# Update processing

- Primary target: DW fact tables
  - Fast bulk insert is critical
  - Reasonable trickle insert/delete/update performance

- Bulk insert
  - Creates row groups directly (if over 1M rows)
- Trickle operations
  - Insert: adds row to delta store
  - Delete: inserts <group id, row no> into B-tree
  - Update: processed as delete + insert

# Performance

- Bulk load rate: measured 600 GB/hour
  - 16 cores, 16 concurrent bulk load streams
- Trickle load rates (single threaded)
  - Single row/transaction: 2,944 rows/sec
  - 1000 rows/transaction: 34,129 rows/sec
- Delta stores transparently included in scans
  - Minimal effect on performance – too small to matter

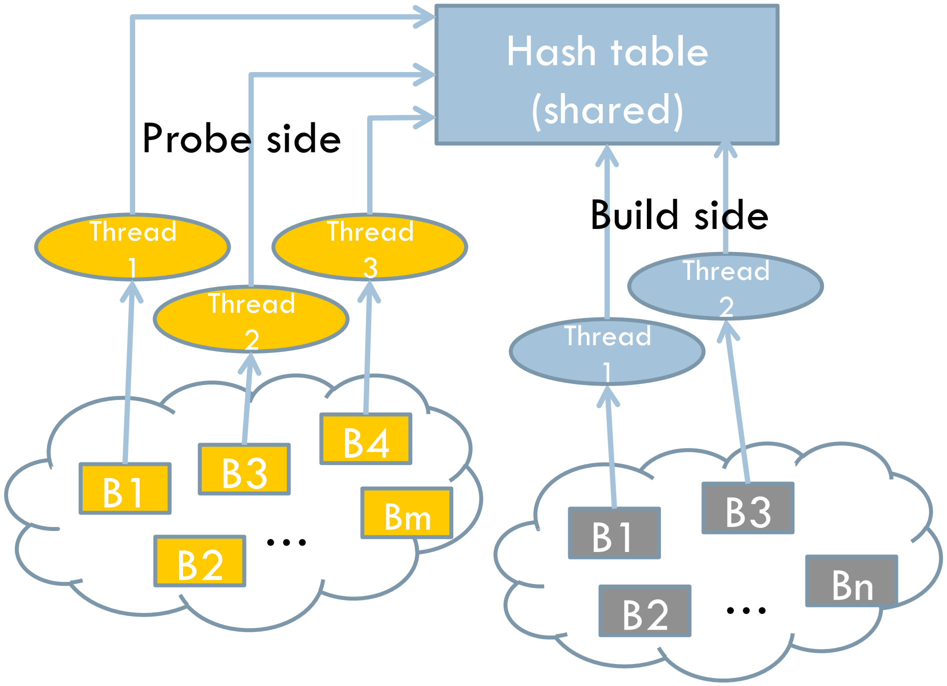
# Outline

- □ Column store indexes in SQL Server 2012
- □ Updatable clustered column store
- □ Query processing enhancements
- □ Archival compression
- □ Performance examples
- □ Status

# Query processing enhancements

- Many improvements to batch hash join
- Improvements to batch hash aggregation
  - Spilling to disk – can be used for final aggregation
- Additional batch mode operators
  - Scalar aggregation, union all
- Batch mode operators can be used anywhere in query plan
  - Decision integrated into optimization process

# Batch mode hash join

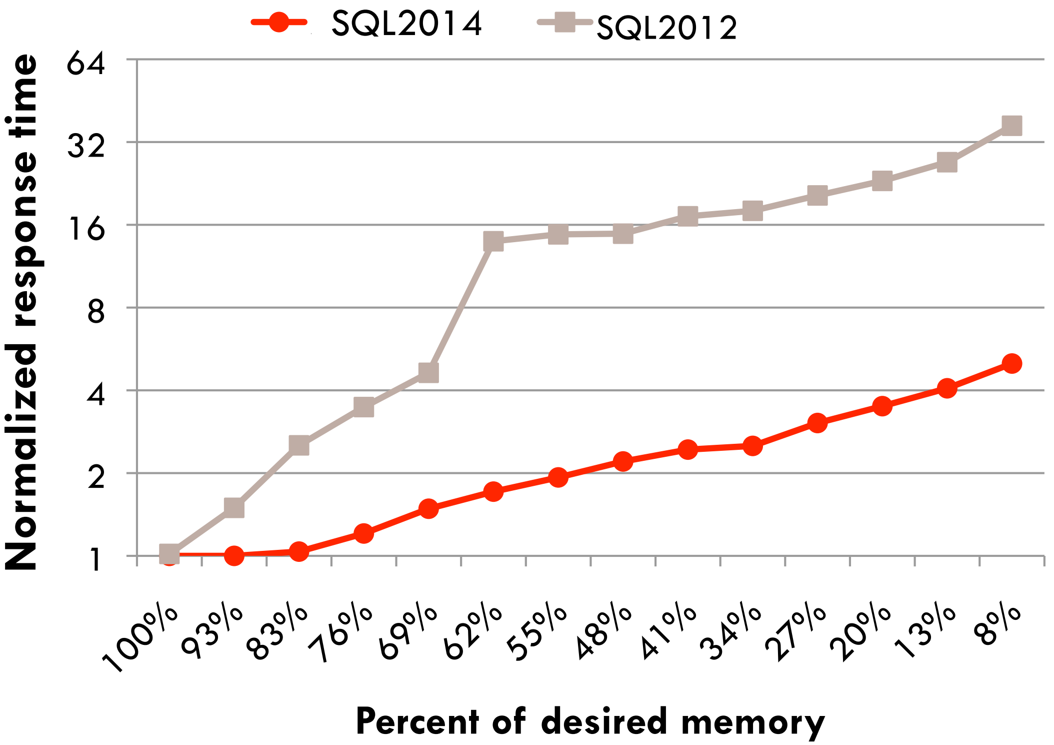


## Enhancements

- All join types
  - Inner, outer, semi, antisemi, cross
- Spill hash table(s) to disk
  - Smart selection of what to spill
- Improvements to bitmap filters

- No repartitioning needed
- Data skew speeds up processing!

# Join performance under memory pressure



Sigmod 2013

# Outline

- Column store indexes in SQL Server 2012
- Updatable clustered column store
- Query processing enhancements
- Archival compression
- Performance examples
- Status

# Archival compression

- □ Further compress on-disk column segments
  - Compress on write
  - Decompress on read
- □ Lempel-Ziv compression algorithm (LZ77)

| Database Name | Raw data size (GB) | Compression ratio | | |
|---|---|---|---|---|
| | | Archival compression? | | GZIP |
| | | No | Yes | |
| EDW | 95.4 | 5.84 | 9.33 | 4.85 |
| Sim | 41.3 | 2.2 | 3.65 | 3.08 |
| Telco | 47.1 | 3.0 | 5.27 | 5.1 |
| SQM | 1.3 | 5.41 | 10.37 | 8.07 |
| MS Sales | 14.7 | 6.92 | 16.11 | 11.93 |
| Hospitality | 1.0 | 23.8 | 70.4 | 43.3 |

# Outline

- □ Column store indexes in SQL Server 2012
- □ Updatable clustered column store
- □ Query processing enhancements
- □ Archival compression
- □ Performance examples
- □ Status

# Query performance

| Query | Rowstore | | Columnstore | | Speedup | | |
|---|---|---|---|---|---|---|---|
| | Cold | Warm | Cold | Warm | **Cold** | **Warm** | |
| Q_count | 13.0 | 4.33 | 0.309 | 0.109 | **42.1** | **39.7** | Count all rows |
| Q_outer | 263 | 1.03 | 4.1 | 0.493 | **64.1** | **2.1** | Filter, left outer join, group-by |
| Q_union_all | 20.8 | 19.0 | 3.0 | 1.41 | **6.9** | **13.5** | Union all, filter, join, group-by |
| Q_count_in | 62.5 | 24.0 | 2.29 | 1.15 | **27.3** | **20.9** | IN predicate, count |
| Q_not_in | 12.0 | 10.2 | 6.95 | 1.31 | **1.7** | **7.8** | NOT IN subquery, group-by |

- TPC-E database
  - store_sales with 288M rows plus smaller dimension tables
- Machine: 16 cores, 48 GB, 4 disks

# Current status

- Already shipping in SQL Server PDW V2 (Parallel Data Warehouse)
- Will ship in SQL Server 2014
  - First public beta by the end of June
  - To be released late this year

Microsoft®

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.