Column Store Internals

Sebastian Meine
SQL Stylist with sqlity.net
sebastian@sqlity.net

Outline

- Column Store
- Storage
- Aggregates
- Batch Processing
### History

<table>
<thead>
<tr>
<th>Year</th>
<th>Event Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>First mention of idea to cluster column groups into separate files</td>
<td>[J. A. Hoffer, D. G. Severance]</td>
</tr>
<tr>
<td>1985</td>
<td>First suggestion of fully decomposed storage</td>
<td>[G. P. Copeland and S. Khoshafian]</td>
</tr>
<tr>
<td>1996</td>
<td>First commercial columnar database: Sybase IQ</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>First general-purpose DBMS to fully integrate columnar storage and processing: SQL Server 2012</td>
<td></td>
</tr>
</tbody>
</table>

Source: [Larson et al]

### HoBT

![HoBT Diagram](image)
Column Store

Forms of Storage

<table>
<thead>
<tr>
<th>NSM</th>
<th>DSM</th>
<th>PAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>• N-ary Storage Model</td>
<td>• Decomposition Storage Model</td>
<td>• Partition Attributes Across (Ailamaki &amp; DeWitt, 2001)</td>
</tr>
</tbody>
</table>
xVelocity

**xVelocity In-Memory Analytics Engine**
- SQL Server Analysis Services
- PowerPivot

**xVelocity Memory-Optimized Columnstore Index**
- SQL Server Database Engine

---

**xVelocity Memory-Optimized Columnstore Index**

- Not an “in-memory” construct
- Columns stored independently
- Uses VertiPaq™ compression
- Requires Enterprise Edition
Not an Index

- No Order
- No Key
- Not a bitmap index

Segment

- ~ 1 million rows
- Aligned between columns
- Base Table Order preserved

- Each column
- Stored in one continuous BLOB
- Independently compressed
Compression

VertiPaq™

- Proprietary
- Not Documented

- Dictionary Encoding
- Huffman Encoding
- Run Length Encoding
- Lempel-Ziv-Welch

Partitioning

- Fully supported
- Must be aligned to base table
  - Must include partition column
- Allows for trickle load
Redirect: BLOBs

- Separate Allocation Unit
- Pages $\infty \leftrightarrow \infty$ Values
- Modified B+Tree per Value

Structure

- Columns
- BLOBs
- Segments
Dictionaries

- Stored in Separate BLOB
- Per Partition and Column Primary
- Per multiple Segments Secondary or Shared
- Not shared between Columns or Partitions

Creation

<table>
<thead>
<tr>
<th>Memory requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>• (4.2 * #Cols + 68) * DOP + 34 * #StringCols MBs</td>
</tr>
<tr>
<td>• Might cause Msg 701,802, 8657 or 8658</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rows per segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>• N threads -&gt; N smaller segments</td>
</tr>
<tr>
<td>• Parallelism only for &gt; 10⁶ rows</td>
</tr>
</tbody>
</table>
Cache

New cache design
Ensures contiguous storage of segments in memory
Cached on a segment basis
Can handle free memory < index size

RBAR
Row By Agonizing Row
2007
Jeff Moden
Relational Operator (RelOp)

- Clustered Index Scan
  - Iterator
    - open
    - getRow
    - close

Relational Operator (RelOp)

- Columnstore Index Scan
  - Iterator
    - open
    - getRow
    - getBatch
    - close
Batch Processing

- Only on Columnstore Data
- ~1000 Rows
- Independent Column-Vectors
- Data stays Compressed
- Never Serial

Batch-Advantage

- Loop unrolling
- Memory prefetching
- Branch prediction
- Reduced cache misses
- Reduced TLB misses
**Batchables**

- Scan
- Filter
- Inner hash join
- Batch hash table build
- Local hash (partial) aggregation

---

**Apollo**

*Enterprise Edition Only*

- xVelocity m.o.c.i.
- Vector-based query execution
Segment Elimination

Column-Segment stores
- Actual Min Value
- Actual Max Value

Filter out entire Segments
- Column Filter
- Bitmap Filter

Limitations
- No updates (Partition switching possible)
- Cannot be a clustered index
- Restricted set of batch mode operators
- Restricted join operations
- No filtered columnstore index
- No computed columns
- Not supported on [indexed] views
- Only one per table
- Max 1024 columns
- Cannot include sparse columns
- Cannot enforce primary key or unique constraint
- Cannot be "ALTER INDEX"ed
- Cannot "INCLUDE" columns
- No sort order
- No seek!
- No page or row compression, no vardecimal data format
- No replication, change tracking, CDC (because read only?)
- Only 21 of 36 Data Types
- Situation will be improved in future versions
Best Practices: DOs

- Include all columns
- Favor star-joins, aggregations, and grouping
- Put CS index on large tables (Fact & Dim)
- Prefer small data types

Best Practices: DON’Ts

- Large (mostly) unique string value columns
- UNION ALL of table with and table without columnstore
- Avoid filters and joins on string columns
- Avoid OUTER JOIN and NOT IN
Literature

  - Per-Ake Larson, Eric N. Hanson, Susan L. Price
- [Abadi et al] Column-Stores vs. Row-Stores: How different are they really? (2008, SIGMOD)
  - Daniel L. Abadi, Samuel R. Madden, Nabil Hachem
- SQL Server Columnstore Index FAQ (microsoft.com)
- SQL Server Columnstore Performance Tuning (microsoft.com)
- [Campbell] The coming in-memory database tipping point (2012, blogs.technet.com)
  - David Campbell
- Perform Scalar Aggregates and still get the Benefit of Batch Processing (microsoft.com)
- Work Around Performance Issues for Columnstores Related to Strings (microsoft.com)
- Ensuring Your Data is Sorted or Nearly Sorted by Date to Benefit from Date Range Elimination (microsoft.com)
- Columnstore Indexes (msdn.microsoft.com)
- [Rusanu] Inside the SQL Server 2012 Columnstore Index (2012, rusanu.com)
  - Rusanu Consulting llc
- Multi-Dimensional Clustering to Maximize the Benefit of Segment Elimination (microsoft.com)

References

SQL Stylist with sqlity.net
sebastian@sqlity.net

http://goo.gl/XnmRg

Sebastian Meine

Session Materials