ColumnStore Index in SQL Server 2016 and Azure SQL Database
and quick overview of Operational Analytics

Assaf Fraenkel
Senior Architect
Microsoft Consulting Services
Session Objectives And Takeaways

Session Objective(s):
Rationale for using columnstore index for DW
Showcase new enhancements to columnstore index in SQL AzureDB and SQL 2016
Rich set of configurations for Real-Time Analytics
First leading commercial database combining in-memory OLTP and in-memory DW

Key Takeaways
Significant improvements for DW query performance in SQL AzureDB and SQL 2016
Scale-out Analytics with AlwaysOn configuration
Enhancements in Higher Concurrency, Supportability and data load
I can do real-time analytics with SQL Server with no application changes
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrency and High Availability
Operational Analytics
Demo: Basic Columnstore

Assaf Fraenkel
### World Record Breaking Performance (TPC-H)

**SQL Server 2016 gives 40% improved performance over SQL Server 2014**

---

<table>
<thead>
<tr>
<th>Rank</th>
<th>Company</th>
<th>System</th>
<th>QphH</th>
<th>Price/QphH</th>
<th>Watts/KQphH</th>
<th>System Availability</th>
<th>Database</th>
<th>Operating System</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lenovo System X3850 X6</td>
<td>969,504</td>
<td>.72 USD</td>
<td>NR</td>
<td>NR</td>
<td>07/31/16</td>
<td>Microsoft SQL Server 2016 Enterprise Edition</td>
<td>Microsoft Windows Server 2012 R2 Standard Edition</td>
<td>03/09/16</td>
</tr>
<tr>
<td>4</td>
<td>Hewlett Packard Enterprise</td>
<td>637,000</td>
<td>1.13 USD</td>
<td>NR</td>
<td>NR</td>
<td>03/12/15</td>
<td>Microsoft SQL Server 2014 Enterprise Edition</td>
<td>Microsoft Windows Server 2012 R2 Standard Edition</td>
<td>03/12/15</td>
</tr>
</tbody>
</table>
DW Architecture - Traditional

- **MOLAP**
  - Data Latency
  - Reports limited to Cubes
  - Data Refresh Periodically

- **Vertipaq**
  - Data Latency
  - Data Refresh Periodically

**ETL**

**BI and analytics**
- Dashboards
- Reporting

**SSAS 2016**
- Multi-Dimensional (MOLAP)
- Query Exec

**SQL Server Relational DW**

**Database**

**Query Exec**
- Tabular (VertiPaq)
- Data Refresh

**Hourly, Daily, Weekly**
DW Architecture with SQL 2016

- Key Value Proposition
  - Interactive Analytics Querying
  - Reduced complexity
  - Significant Improvement in Tabular (Direct Query)

ETL

Multi-Dimensional

Tabular

BI and analytics
  - Dashboards
  - Reporting

SSAS 2016

SQL Server Relational DW (columnStore)

Database

Hourly, Daily, Weekly

ROLAP

Direct Query

Direct Query

Direct Query
Columnstore Index: Why?

Data stored as rows
Illustration of data stored in rows...

Ideal for OLTP
Efficient operation on small set of rows

Data stored as columns
Illustration of data stored in columns...

Improved compression:
Data from same domain compress better

Improved Performance:
More data fits in memory
Optimized for CPU utilization

Reduced I/O:
Fetch only columns needed

Ideal for DW Workload
## Columnstore Index Example

<table>
<thead>
<tr>
<th>OrderDateKey</th>
<th>ProductKey</th>
<th>StoreKey</th>
<th>RegionKey</th>
<th>Quantity</th>
<th>SalesAmount</th>
</tr>
</thead>
<tbody>
<tr>
<td>20101107</td>
<td>106</td>
<td>01</td>
<td>1</td>
<td>6</td>
<td>30.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>04</td>
<td>2</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>109</td>
<td>04</td>
<td>2</td>
<td>2</td>
<td>20.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>03</td>
<td>2</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>106</td>
<td>05</td>
<td>3</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>02</td>
<td>1</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101108</td>
<td>102</td>
<td>02</td>
<td>1</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>03</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101108</td>
<td>109</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101109</td>
<td>103</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>17.00</td>
</tr>
</tbody>
</table>
## Horizontally Partition (Row Groups)

<table>
<thead>
<tr>
<th>OrderDateKey</th>
<th>ProductKey</th>
<th>StoreKey</th>
<th>RegionKey</th>
<th>Quantity</th>
<th>SalesAmount</th>
</tr>
</thead>
<tbody>
<tr>
<td>20101107</td>
<td>106</td>
<td>01</td>
<td>1</td>
<td>6</td>
<td>30.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>04</td>
<td>2</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>109</td>
<td>04</td>
<td>2</td>
<td>2</td>
<td>20.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>03</td>
<td>2</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>106</td>
<td>05</td>
<td>3</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>02</td>
<td>1</td>
<td>5</td>
<td>25.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OrderDateKey</th>
<th>ProductKey</th>
<th>StoreKey</th>
<th>RegionKey</th>
<th>Quantity</th>
<th>SalesAmount</th>
</tr>
</thead>
<tbody>
<tr>
<td>20101108</td>
<td>102</td>
<td>02</td>
<td>1</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>03</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101108</td>
<td>109</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101109</td>
<td>103</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>17.00</td>
</tr>
</tbody>
</table>
### Vertically Partition (Segments)

<table>
<thead>
<tr>
<th>OrderDateKey</th>
<th>ProductKey</th>
<th>StoreKey</th>
<th>RegionKey</th>
<th>Quantity</th>
<th>SalesAmount</th>
</tr>
</thead>
<tbody>
<tr>
<td>20101107</td>
<td>106</td>
<td>01</td>
<td>1</td>
<td>6</td>
<td>30.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>04</td>
<td>2</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>109</td>
<td>04</td>
<td>2</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>03</td>
<td>2</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101107</td>
<td>106</td>
<td>05</td>
<td>3</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>02</td>
<td>1</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>20101108</td>
<td>102</td>
<td>02</td>
<td>1</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>03</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101109</td>
<td>109</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101109</td>
<td>103</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>17.00</td>
</tr>
</tbody>
</table>
### Compress Each Segment

<table>
<thead>
<tr>
<th>OrderDateKey</th>
<th>ProductKey</th>
<th>StoreKey</th>
<th>RegionKey</th>
<th>Quantity</th>
<th>SalesAmount</th>
</tr>
</thead>
<tbody>
<tr>
<td>20101107</td>
<td>106</td>
<td>01</td>
<td>1</td>
<td>6</td>
<td>30.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>04</td>
<td>2</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>109</td>
<td>04</td>
<td>2</td>
<td>2</td>
<td>20.00</td>
</tr>
<tr>
<td>20101107</td>
<td>103</td>
<td>03</td>
<td>2</td>
<td>4</td>
<td>17.00</td>
</tr>
<tr>
<td>20101107</td>
<td>106</td>
<td>05</td>
<td>3</td>
<td>5</td>
<td>20.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>02</td>
<td>1</td>
<td>2</td>
<td>25.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OrderDateKey</th>
<th>ProductKey</th>
<th>StoreKey</th>
<th>RegionKey</th>
<th>Quantity</th>
<th>SalesAmount</th>
</tr>
</thead>
<tbody>
<tr>
<td>20101108</td>
<td>102</td>
<td>02</td>
<td>1</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>20101108</td>
<td>106</td>
<td>03</td>
<td>5</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101108</td>
<td>109</td>
<td>01</td>
<td>1</td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>01</td>
<td>2</td>
<td>4</td>
<td>20.00</td>
</tr>
<tr>
<td>20101109</td>
<td>106</td>
<td>04</td>
<td>2</td>
<td>5</td>
<td>25.00</td>
</tr>
<tr>
<td>20101109</td>
<td>103</td>
<td>04</td>
<td>1</td>
<td>1</td>
<td>17.00</td>
</tr>
</tbody>
</table>

*Some segments will compress more than others*

*Encoding and reordering not shown*
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrency and High Availability
Operational Analytics
Data Loading into Columnstore: Bulk Import

Key Points

- Import from External Files (BCP, Bulk Insert, SSIS)
- Each Bulk Import thread takes X lock on the RG.
- If batchsize < 100k, the rows are inserted into a delta RG otherwise a new compressed RG is generated
- Minimal Logging (only when imported into compressed rowgroup)
- Don’t specify TABLOCK when importing unlike rowstore where it is required

Bulk load rates for clustered column store have been measured at about 600GB/hour on a 16 core machine, using 16 concurrent bulk load jobs (one per core) targeting the same table.
Data Loading into Columnstore: Staging Table

- Load data from External Files from staging table

Key Points:
- Similar to Bulk Import but with single BATCH
- Residual rows < 100k are inserted into delta store
- SQL 2016: Parallel Insert into Columnstore index
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrency and High Availability
Operational Analytics
- Master copy of the data (10 x compression)
- Only index supported. Simplified analytics
- No PK/FK constraints. Uniqueness can be enforced through Materialized Views
- Locking granularity for UPD/DEL at rowgroup level
- DDL – ALTER, REBUILD, REORGANIZE

SQL AzureDB/SQL 2016

- Master copy of the data (10x compression)
- Additional Btree indexes for efficient equality and short-range searches and PK/FK constraints.
- Locking granularity at row level using NCI index path
- DDL – ALTER, REBUILD, REORGANIZE
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrent and High Availability
Operational Analytics
SELECT ProductKey, SUM(SalesAmount) 
FROM SalesTable 
WHERE OrderDateKey < 20101108 
GROUP BY ProductKey
Multi-Row Batch – Batch Mode Processing

Process multiple rows together for efficiency
Significant reduction in instructions

New in SQL 2016
Single Thread Batch Mode Execution
Batch Anti-Semi-Join
Multiple count DISTINCT
Window Aggregates
Sort
Performance: Aggregate Pushdown (SQL2016)

Select \text{SUM(sales)} \text{ from } <\text{columnstore}>

- Performance gains using Fast Path
  - Opportunistic evaluation of aggregation in storage layer
  - Efficient Aggregation on compressed/encoded data in cache-friendly execution (column at a time)
  - Leverages SIMD
  - Datatype $\leq 8$ bytes no strings
  - Aggregates – MIN, MAX, SUM, COUNT, AVG
  - Works with GroupBy
Performance: String Predicate Pushdown (SQL 2016)

### SQL 2014
- **Filter**
  - String Predicate: `name like '%foo%'`
  - 20 rows
- **Scan**
  - 10 million rows (batches)
  - 50 rows

### SQL 2016
- **Filter**
  - 100k rows (batches)
  - 30 rows
- **Scan**
  - 10 million rows
  - 50 rows

#### Execution Strategy
- Retrieve 10 million rows by converting dictionary encoded value to string
- Apply string predicate on 10 million rows

**Filtered Rows:**
- Rows returned by Filter node = (R1 + filtered R2)
Update - Avoid

Try to do in the ETL
Try to do on a partition that will be switched IN
Update only what you need
Try to avoid 😊
Join might be better
Analytics with Window Aggregates (SQL 2016)
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrency and High Availability
Operational Analytics
Supportability: Index Maintenance

<table>
<thead>
<tr>
<th>Operation</th>
<th>SQL 2014</th>
<th>SQL 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removing Deleted Rows</td>
<td>Requires Index REBUILD</td>
<td>Index REORGANIZE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Remove deleted from single compressed RG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Merge one or more compressed RGs with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleted rows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Done ONLINE</td>
</tr>
<tr>
<td>Smaller RG size resulting from</td>
<td>Index REBUILD</td>
<td>Index REORGANIZE</td>
</tr>
<tr>
<td>• Smaller BATCHSIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Memory Pressure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Index build Residual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ordering Rows</td>
<td>• Create clustered index</td>
<td>No Changes</td>
</tr>
<tr>
<td></td>
<td>• Create columnstore index by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dropping clustered Index</td>
<td></td>
</tr>
</tbody>
</table>
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrency and High Availability
Operational Analytics
## Clustersed Columnstore Index: Concurrency

<table>
<thead>
<tr>
<th>Operation</th>
<th>SQL 2014</th>
<th>SQL 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>• Rowgroup granularity</td>
<td>• Support of SI and RCSI (non blocking)</td>
</tr>
<tr>
<td></td>
<td>• No support of RCSI or SI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Recommendation: use Read Uncommitted</td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td>• Lock at row level (trickle insert)</td>
<td>• No changes</td>
</tr>
<tr>
<td></td>
<td>• Rowgroup level for set of rows</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>• Lock at Rowgroup level</td>
<td>• Row level lock in conjunction with NCI</td>
</tr>
<tr>
<td>Update</td>
<td>• Lock at Rowgroup level</td>
<td>• Row level lock in conjunction with NCI</td>
</tr>
<tr>
<td></td>
<td>• Implemented as Delete/Insert</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>SQL 2014</td>
<td>SQL 2016</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
</tbody>
</table>
| Backup/Restore                    | • Just like any other index.  
• Each SEGMENT is persisted using LOB datatype | • No changes                                |
| AlwaysOn Failover Clustering (FCI) | • Fully Supported                                                        | • Fully Supported                            |
| AlwaysON Availability Groups      | • Fully supported except Readable Secondary                              | • Fully Supported with Readable Secondary    |
| Index Create/Rebuild              | • Offline                                                                | • Offline                                    |
Outline

Overview
Data Load
New physical Structures
Query Performance
Supportability
Concurrency and High Availability
Operational Analytics
Key Issues

- Complex Implementation
- Requires two Servers (CapEx and OpEx)
- Data Latency in Analytics
- More businesses demand/require real-time Analytics

```
Insert into <transactions>
values (‘<upc-code>, ‘flowers’, $20.00)

Select ProduceName, ExpiryDate, Quantity, Sum(Sales)
From <transactions>
Where ProduceType = ‘perishables’
Group By ProduceName, ExpiryDate
```
Insert into <transactions> values ('<upc-code>', 'flowers', $20.00)

Select ProduceName, ExpiryDate, Quantity, sum(sales) From <transactions> Where ProduceType = 'perishables' Group By ProduceName, ExpiryDate

- **Benefits**
  - No Data Latency
  - No ETL
  - No Separate DW

- **Challenges**
  - Minimizing Impact on OLTP workload
  - Delivering Performant Analytics

**This is Real-Time ANALYTICS**
Demo: Real Time Analytics - NCCI

Assaf Fraenkel
SQL Real-Time Analytics

Faster Transactions +

IN-MEMORY OLTP

+ 

Up to **30x faster** transaction processing with In-Memory OLTP

---

**In 2016 and Azure DB**

Faster Analytics

IN-MEMORY DW

---

Over **100x analytics query speed** and significant data compression with In-Memory ColumnStore

---

Using the same tables
Please fill online evaluation for both speakers and overall event.

You have both links in the last EVENT UPDATE email:

Session evaluation form:
http://www.sqlsaturday.com/481/sessions/sessionevaluation.aspx

Overall event evaluation form:
Q&A