#sqlpass

Real-Time Operational Analytics

Erma van Alebeek, Manon van den Burg, Rob Pellicaan |16 juni 2016| Info Support Veenendaal

InfoSupport
Solid Innovator

Content

- Real-Time Operational Analytics an Introduction
- Column Store Indexes
- Real-Time Analytics on disk-based OLTP
- Real-Time Analytics on in-memory OLTP

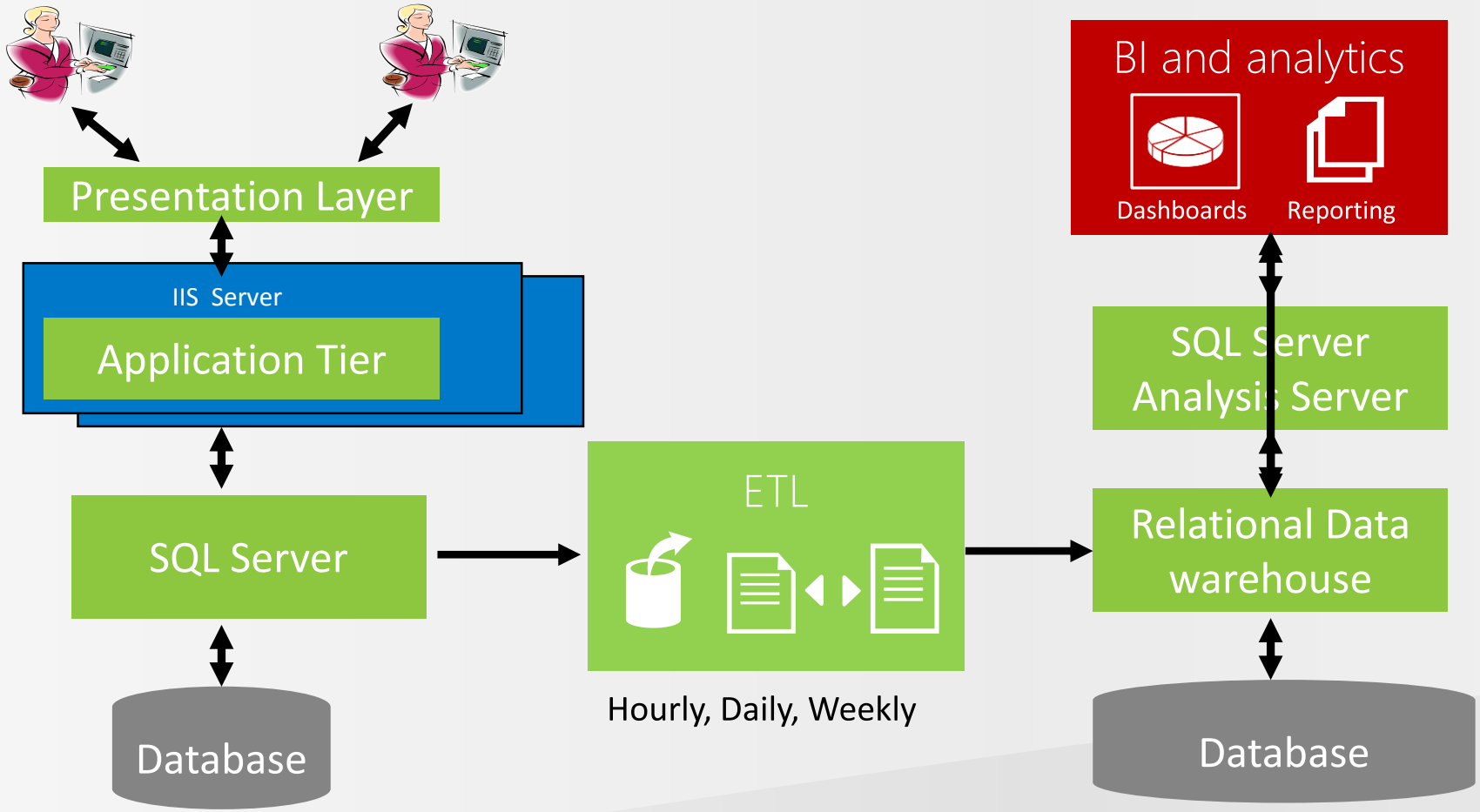
Demo

infoSupport Solid Innovator

Real-Time Operational Analytics

InfoSupport
Solid Innovator

Traditional BI/Analytics architecture



Key Issues

- Complex implementation
- Requires two servers
- Data latency in analytics

- More businesses demand/require real-time analytics

Minimizing Data Latency for Analytics

Presentation Layer

IIS Server

Application Tier

SQL Server

Database

BI and analytics

Dashboards Reporting

SQL Server Analysis Server

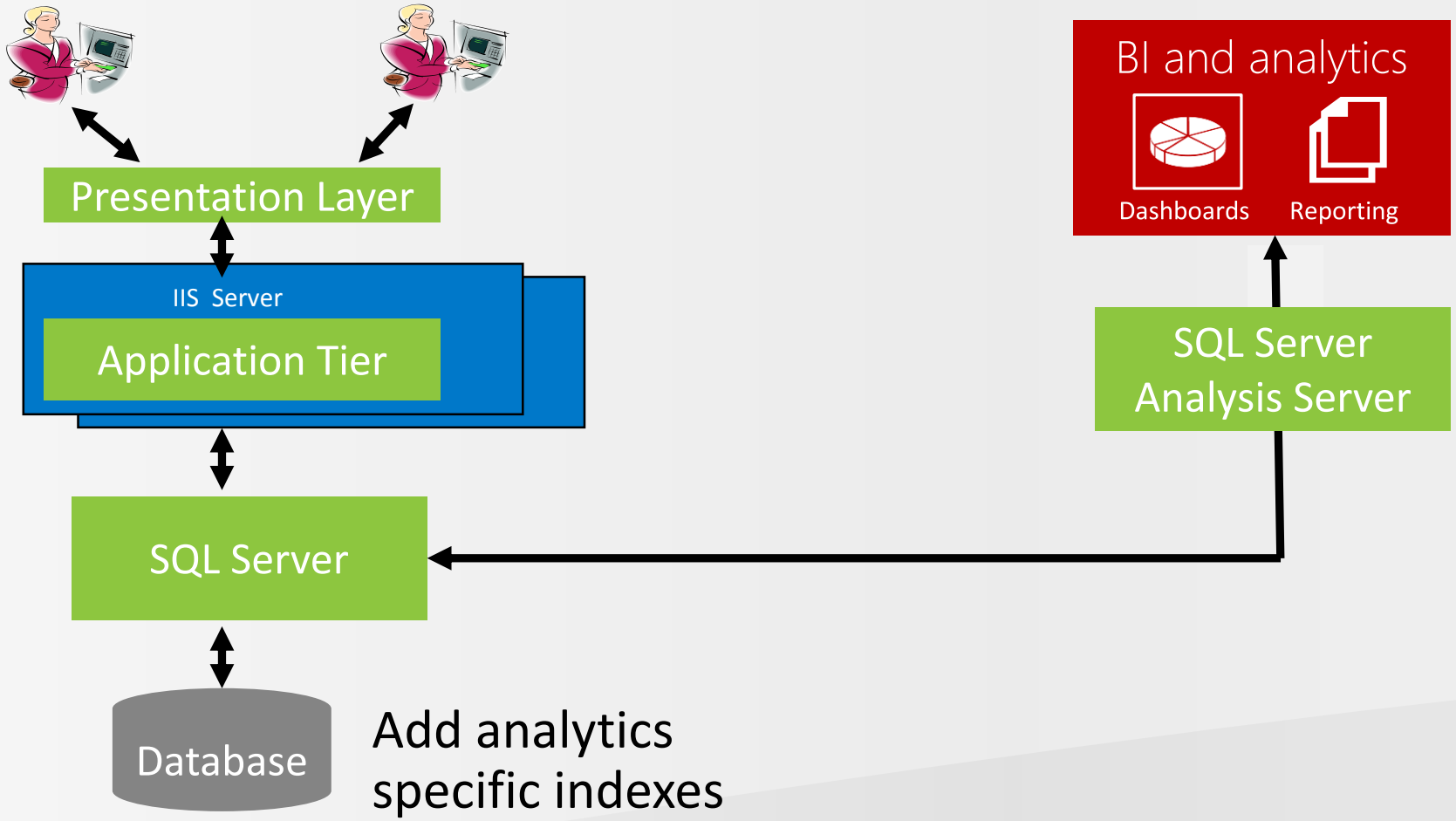
This is OPERATIONAL ANALYTICS

Add analytics specific indexes

Benefits

- No Data Latency
- No ETL
- No separate Data Warehouse

Minimizing Data Latency for Analytics



Add analytics specific indexes

Benefits

- No Data Latency
- No ETL
- No separate Data Warehouse

Challenges

- Analytics queries are resource intensive and can cause blocking
- How to minimize impact on operational workload
- Sub-optimal execution of analytics on relational schema

Goal

Performant analytics on operational schema

Minimal impact on operational workload

Achieved using columnstore Index

Operational Analytics

Ability to run analytics queries concurrently with operational workload using the same schema

Not a replacement for:

- Extreme analytics queries performance possible using schemas customized (e.g. star/snowflake) and pre-aggregated cubes
- Data coming from non-relational sources
- Data coming from multiple relational sources requiring integrated analytic

Column Store Indexes

Recap and enhancements

Rowstore vs Columnstore

Rowstore

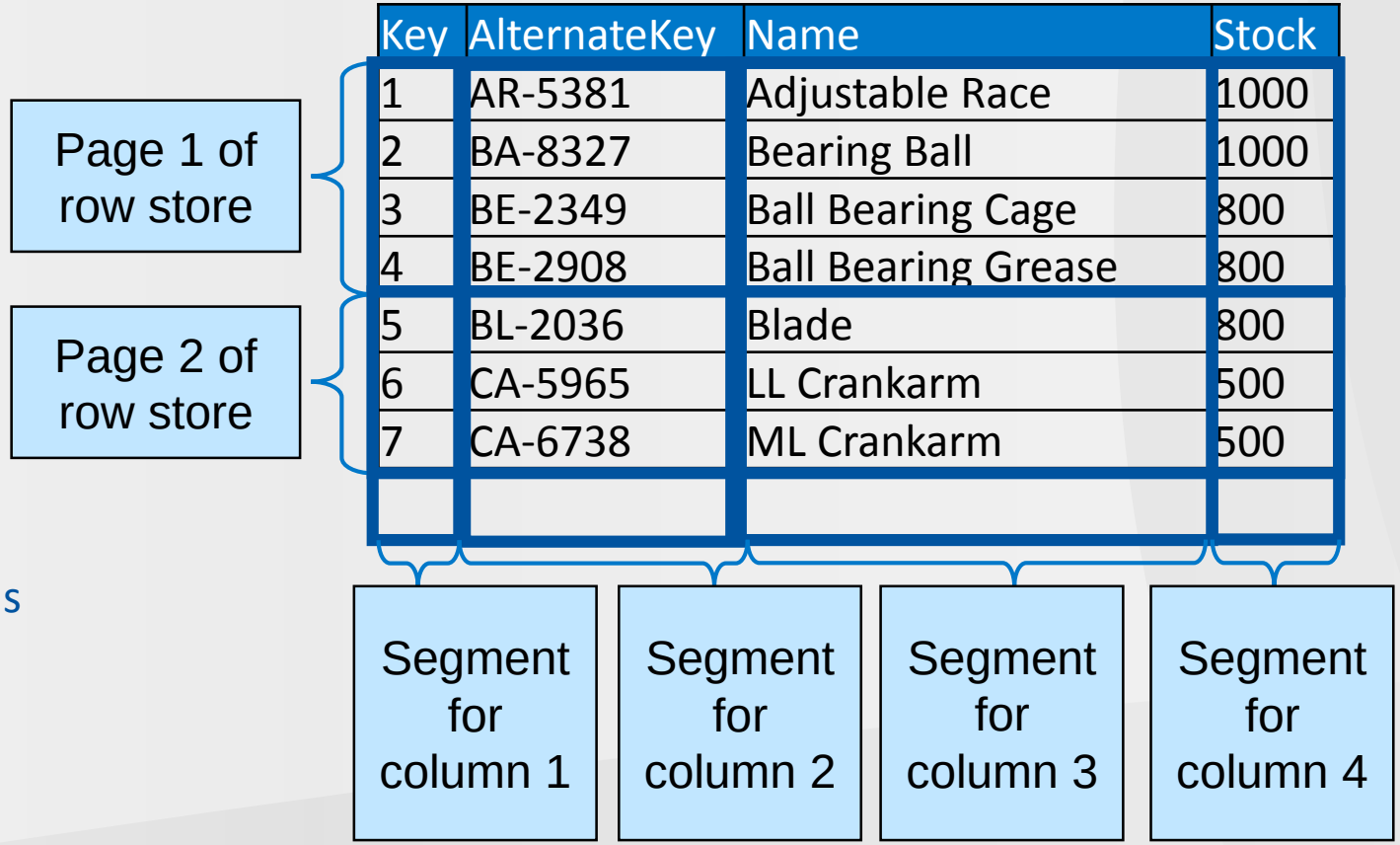
Data stored in a sequence of rows

Good for single row access, equality searches

Columnstore

Data stored in a sequence of columns

Good for aggregate queries



Columnstore index features

Improved compression

Data from same domain compress better

Reduced I/O

Fetch only columns needed

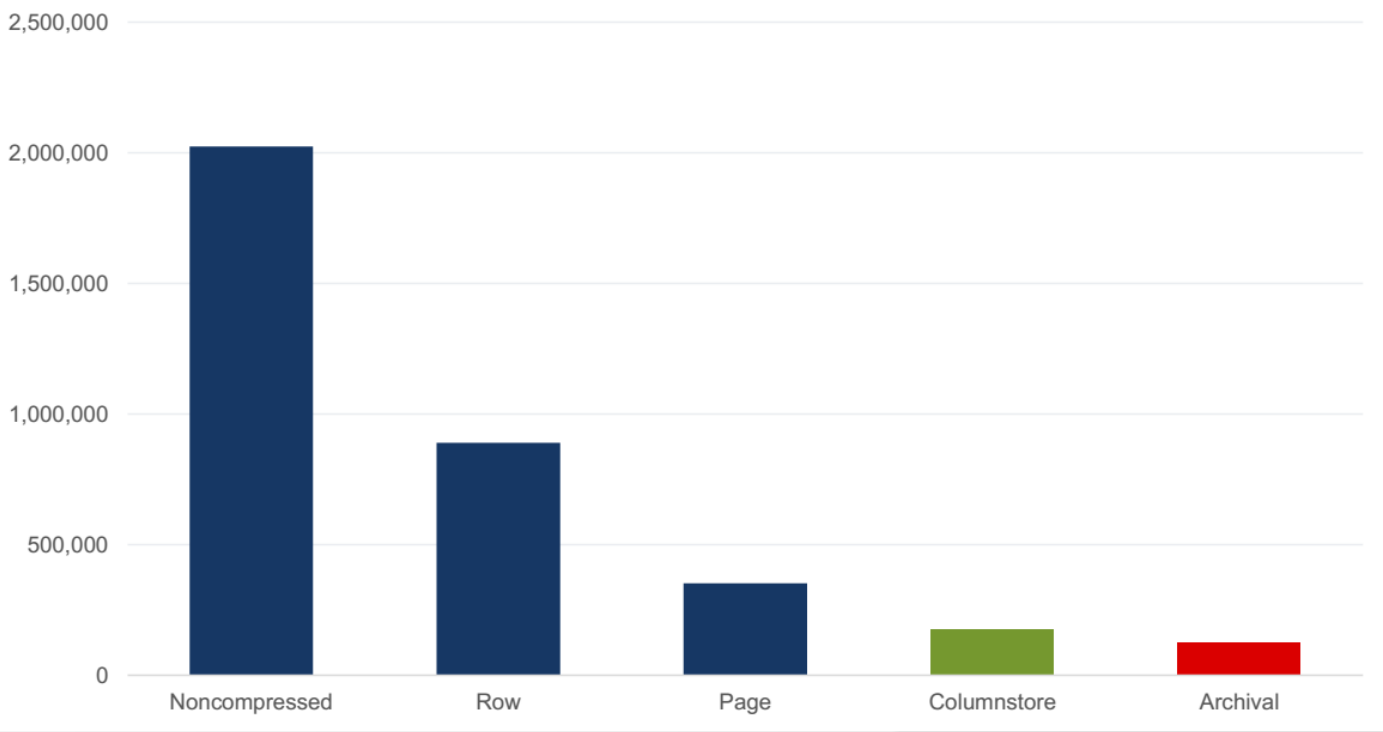
Improved performance

More data fits in memory

Optimized for CPU utilization

Ideal for Data Warehouse workload

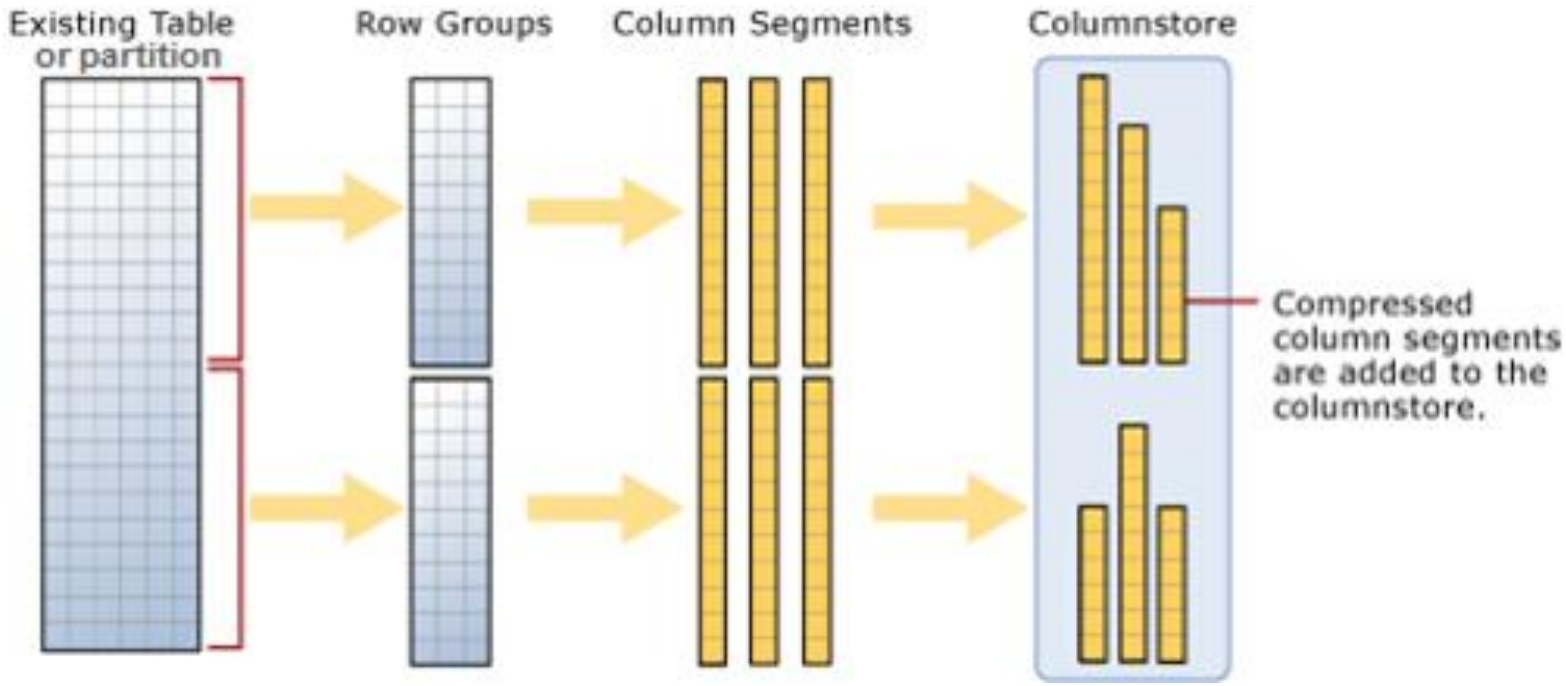
Improved Compression



- Value Scale
- Bit-Array
- Run-length compression
- Dictionary encoding
- Hufman encoding
- Binary compression

Columnstore index

rowgroups and columnsegments



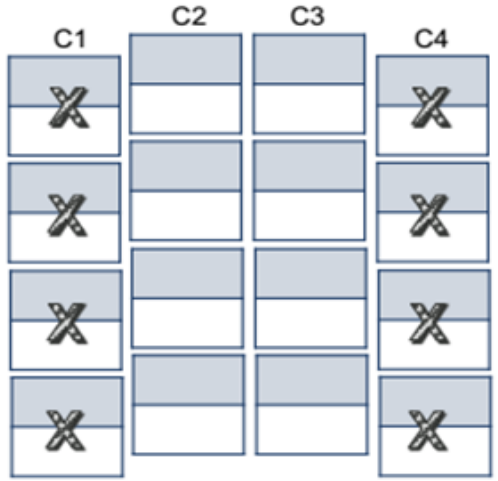
#rows per row group
<= 1,048,576 rows

One column segment for every column
Each segment is compressed together

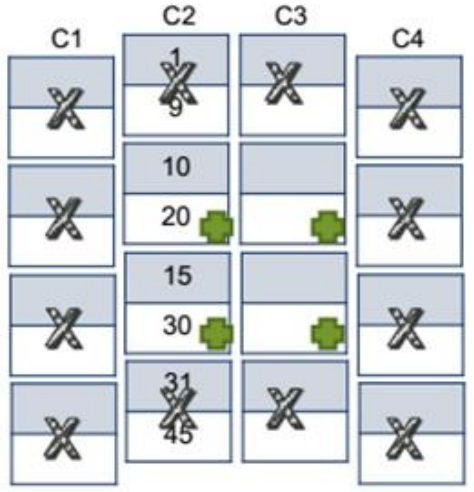
Reading from a columnstore index

```
SELECT SUM(c2), SUM(c3)
FROM dbo.FactOnlineSales
WHERE c2 > 10 AND c2 <=20
```

Data stored column by column
→ column elimination

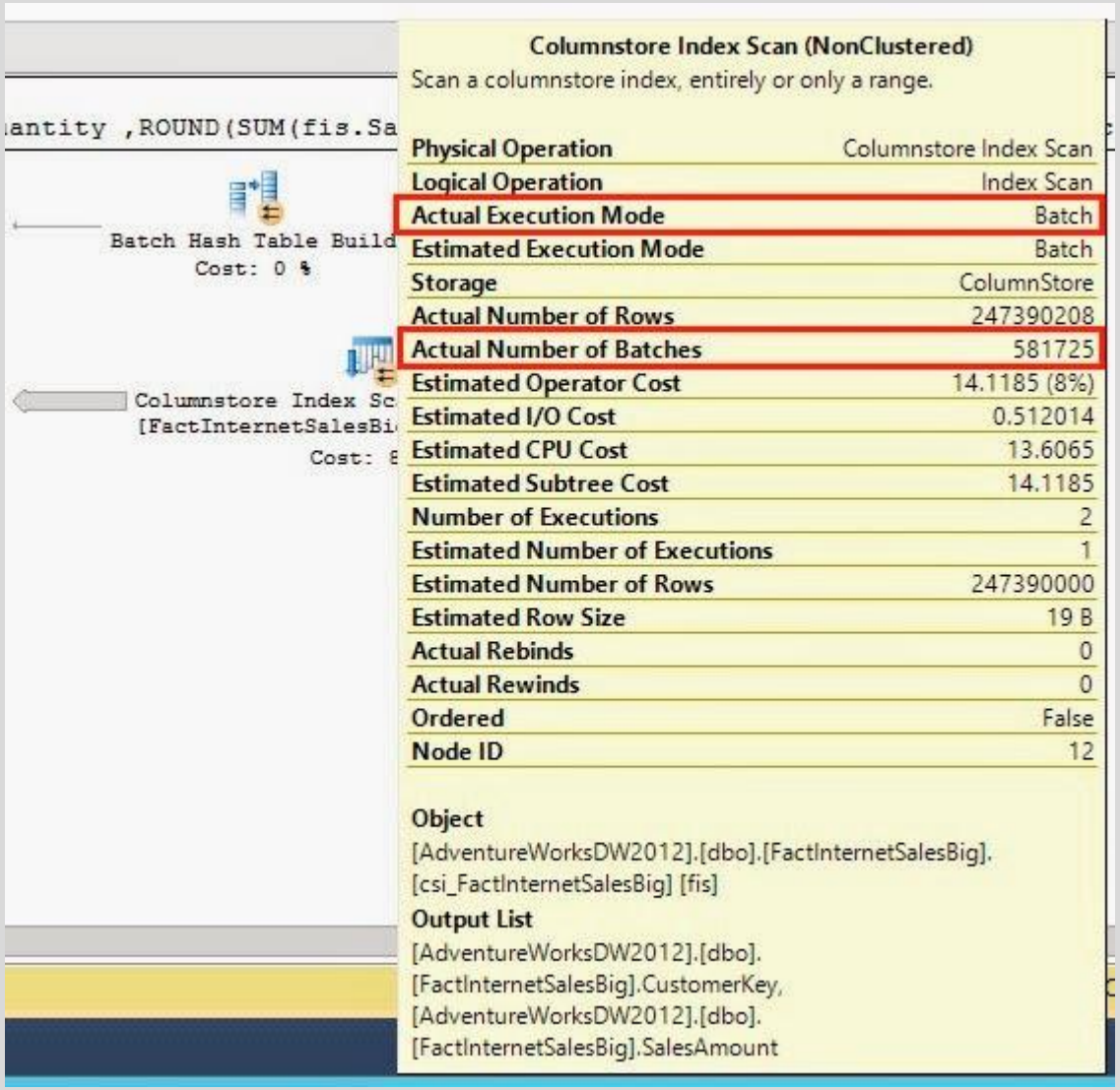


Min and max value stored per rowgroup
→ segment elimination



Batch Execution

- allows the CPU to process approximately 1000 rows at a time
- Special CPU instructions (SIMD)
- Faster query processing, lower CPU utilization



Columnstore indexes

	Clustered	Nonclustered
Disk Based	• Primary storage (=data) Nonclustered B-tree indexes can be added since 2016 !!	• Copy of data • Subset or all rows and columns Updateable since 2016 !!

enables real-time operational Analytics !!

	Clustered	Nonclustered
Memory Based	• Full copy of data (all columns all rows) Since 2016 !!	Not Available (yet)

enables real-time operational Analytics !!

- Share the same storage format
- Records are NOT sorted in either index !!

#sqlpass

Real-Time Analytics on Disk-Based OLTP

InfoSupport
Solid Innovator

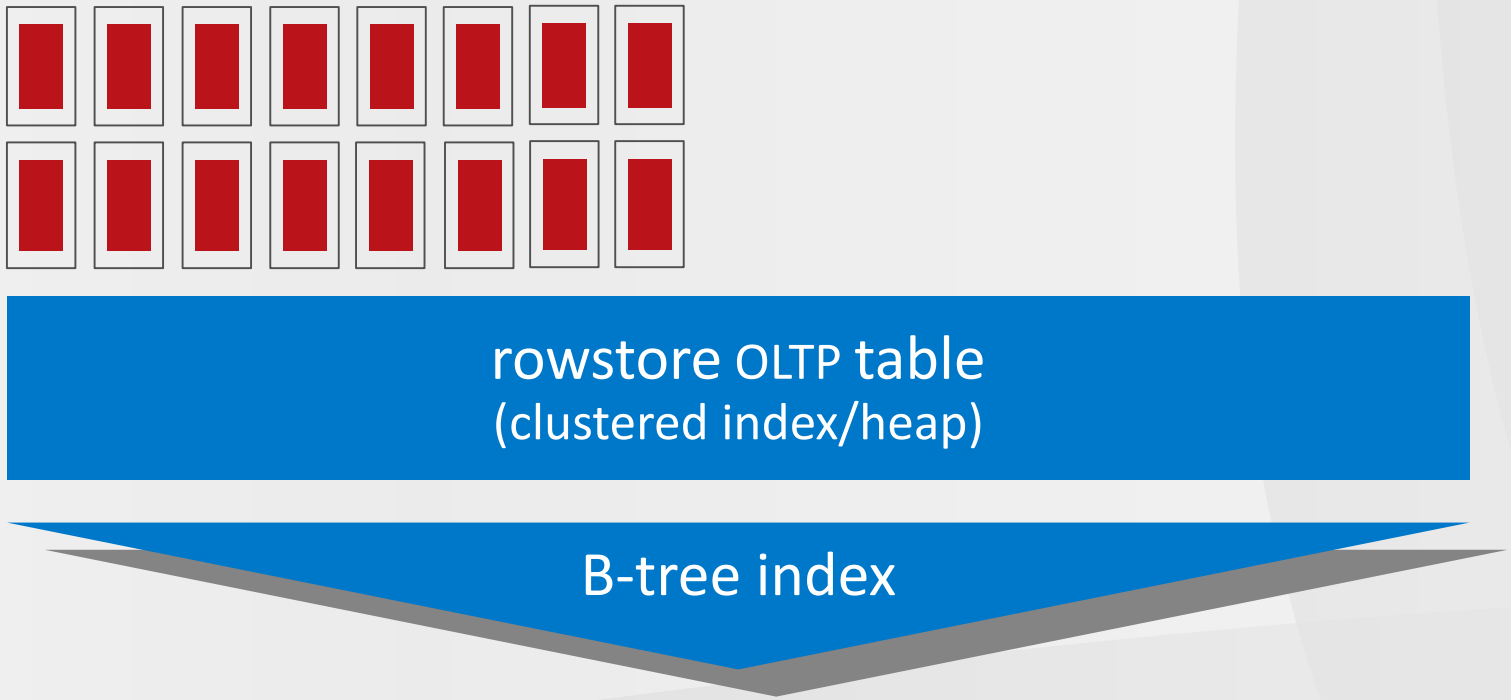
SQL Server 2012 introduced read-only nonclustered columnstore indexes

Usability

- Data warehouse

Key Issues

- Redundancy
- Read-only



SQL Server 2014 introduced updateable clustered columnstore indexes

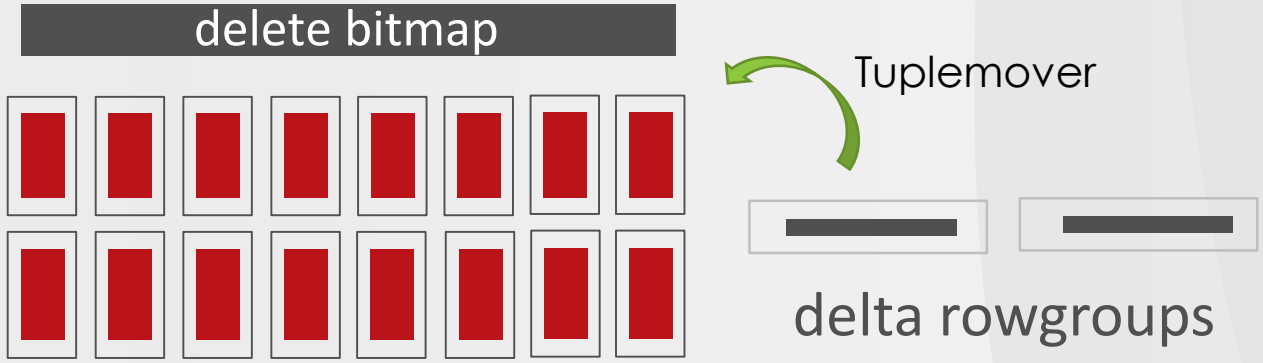
Implemented through delete bitmap and delta rowgroups

Usability

- Data warehouse

Key Issues

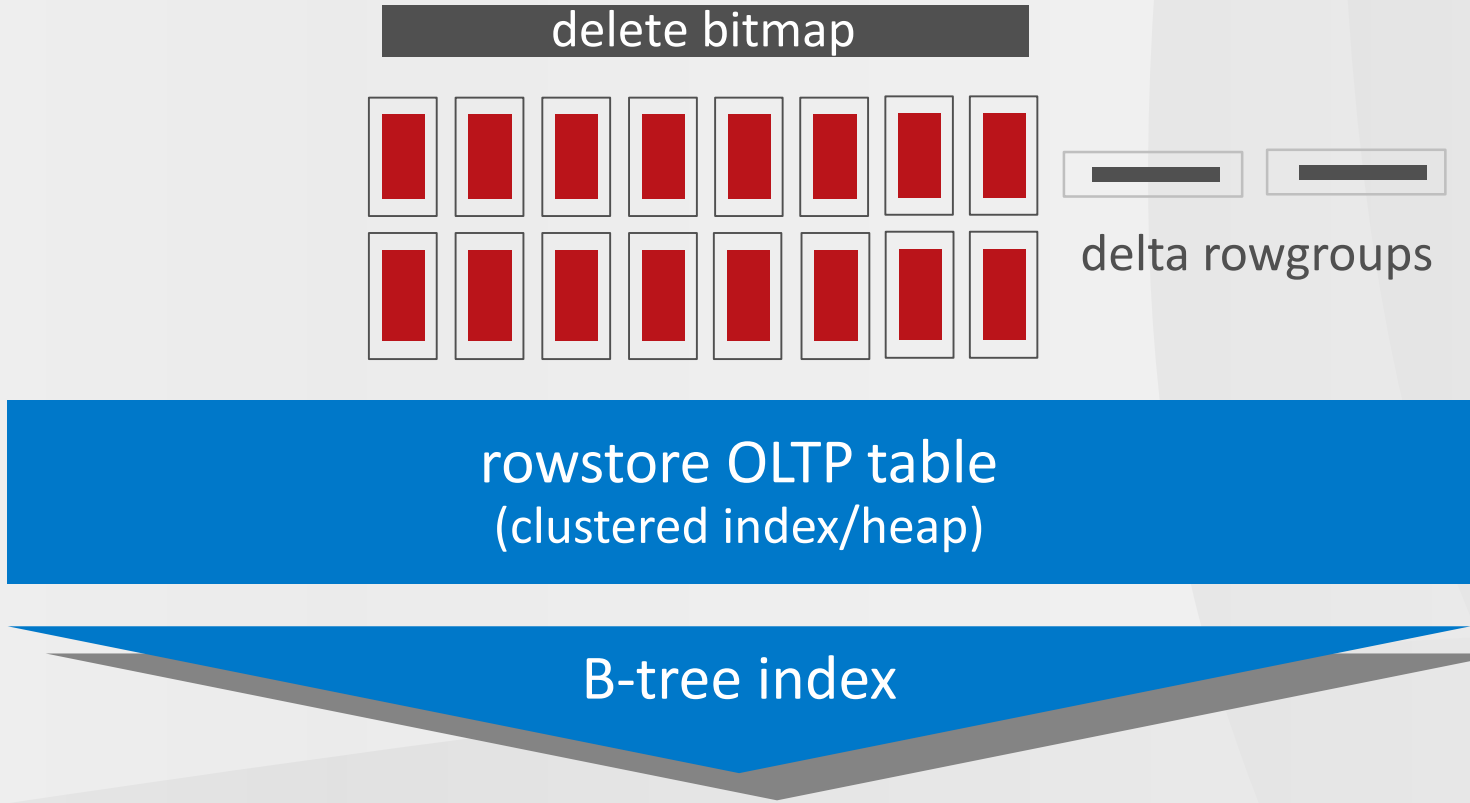
- No additional indexes
- No referential integrity
- Slow for trickle inserts



SQL Server 2016 introduces updateable nonclustered columnstore indexes

Usability

- Real-Time Operational Analytics

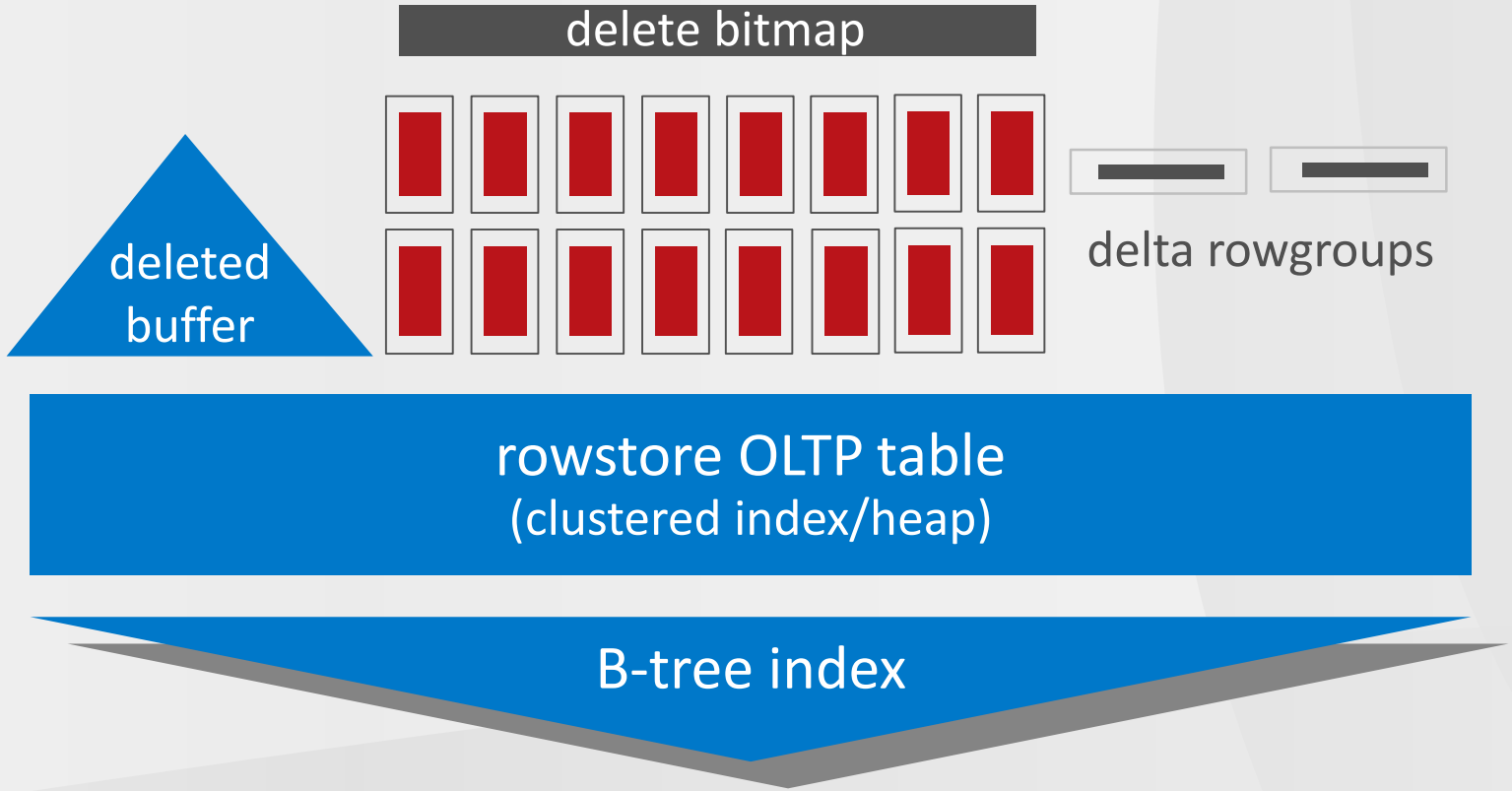


SQL Server 2016 introduces updateable nonclustered columnstore indexes

Implemented through deleted buffer

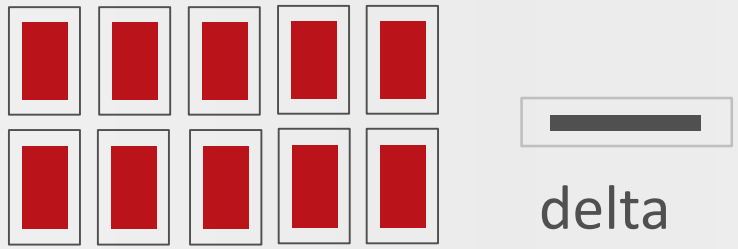
Deleted buffer

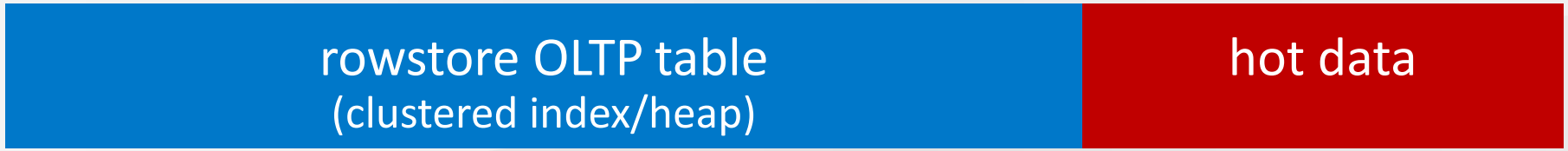
- Contains references to the rows that are deleted/updated but are not yet copied into the deleted bitmap



SQL Server 2016 – Filtered Index

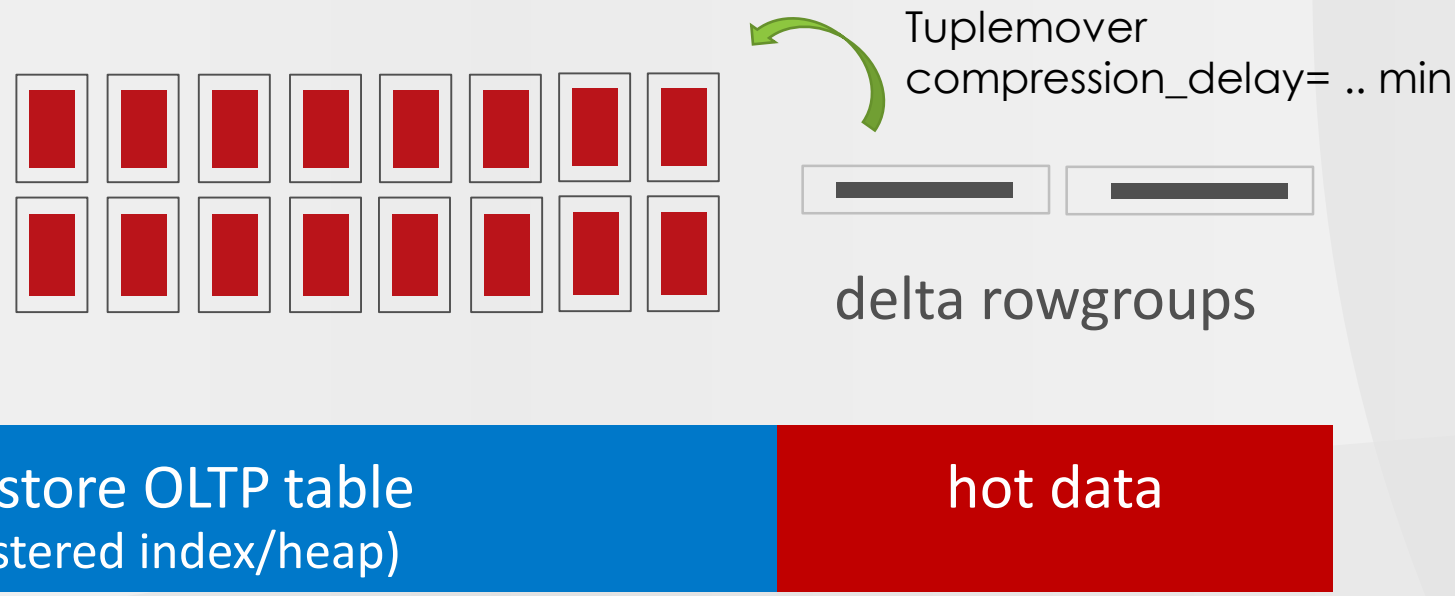
- to index only cold data
- analytics queries access both column store and hot data transparently





SQL Server 2016 – Compression Delay

- to delay data compression of the Delta-Stores (open and the closed ones)



#sqlpass

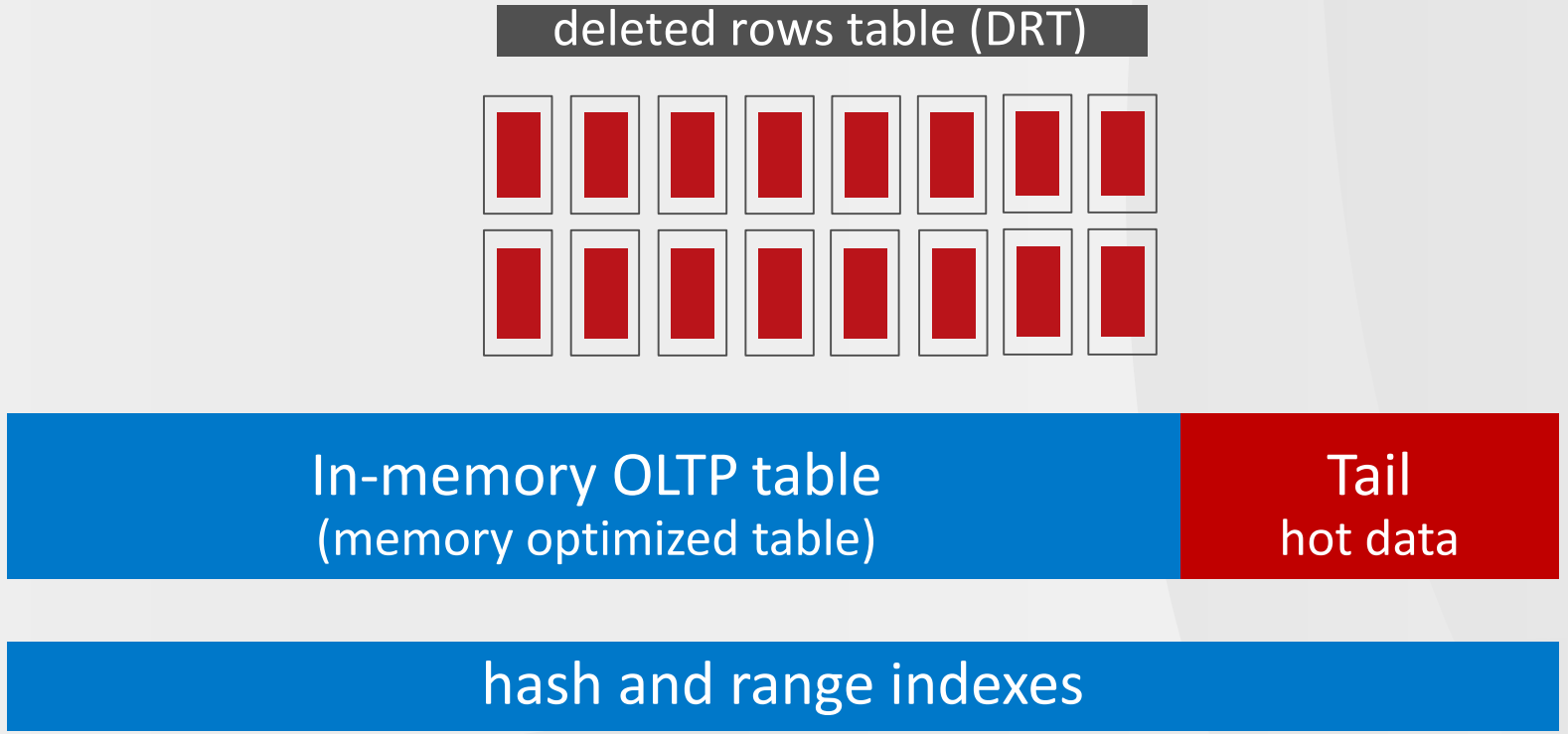
Real-Time Analytics on In-Memory OLTP

InfoSupport
Solid Innovator

SQL Server 2016 introduces clustered columnstore indexes on in-memory tables

In-memory clustered columnstore index

- Full *copy* of data
  (all rows, all columns)
- Completely In-Memory



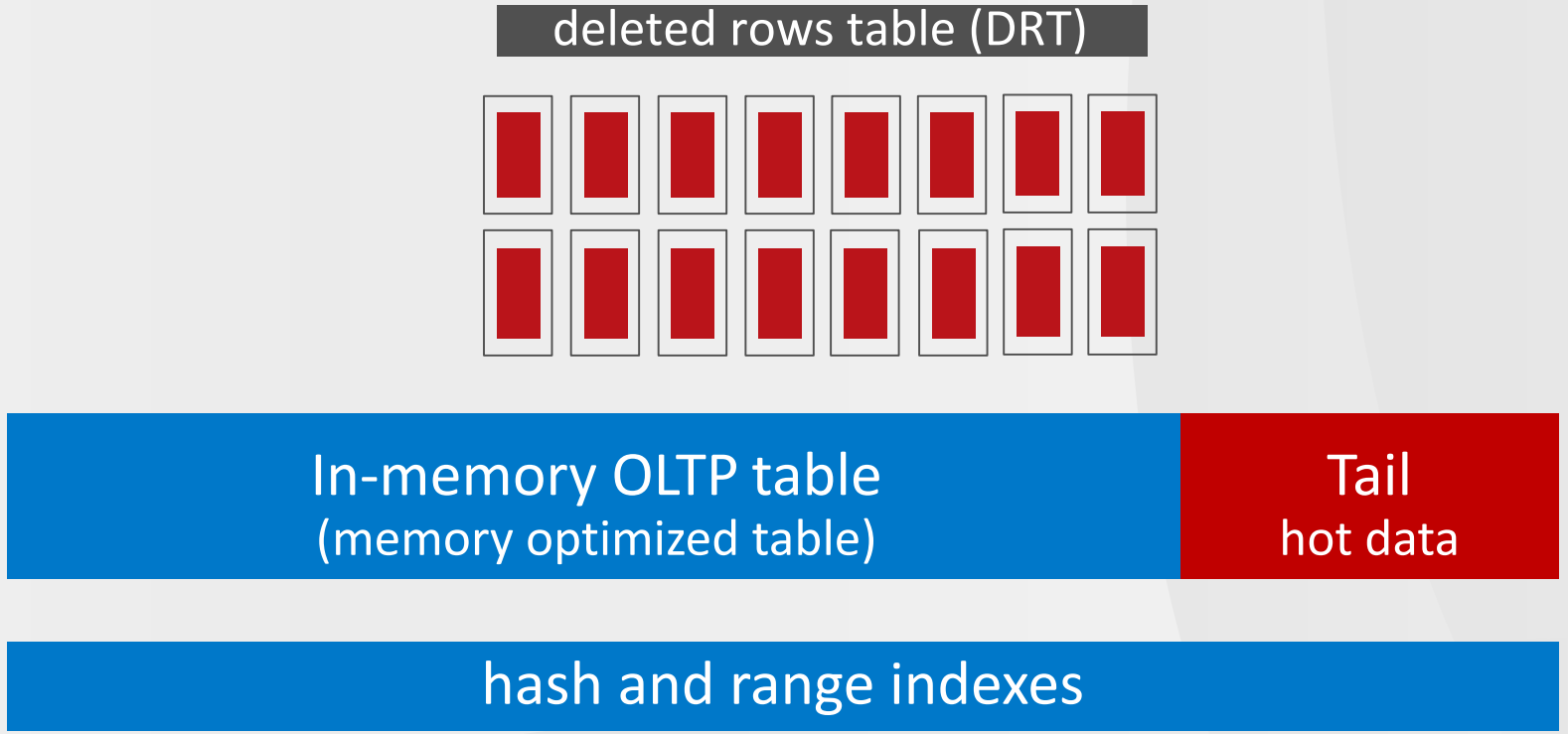
SQL Server 2016 introduces clustered columnstore indexes on in-memory tables

Use of in-memory table tail instead of delta rowgroups

- Rows are moved if not modified in the past hour in chunks of <= 1045678 rows.
- Rows receive a *Columnstore RowId* when compressed *(*rowgroup and rowposition)
- Compression Delay can be used

Deleted Rows Table

- Hash-based In-Memory table
- stores the *Columnstore RowId's* of modified rows



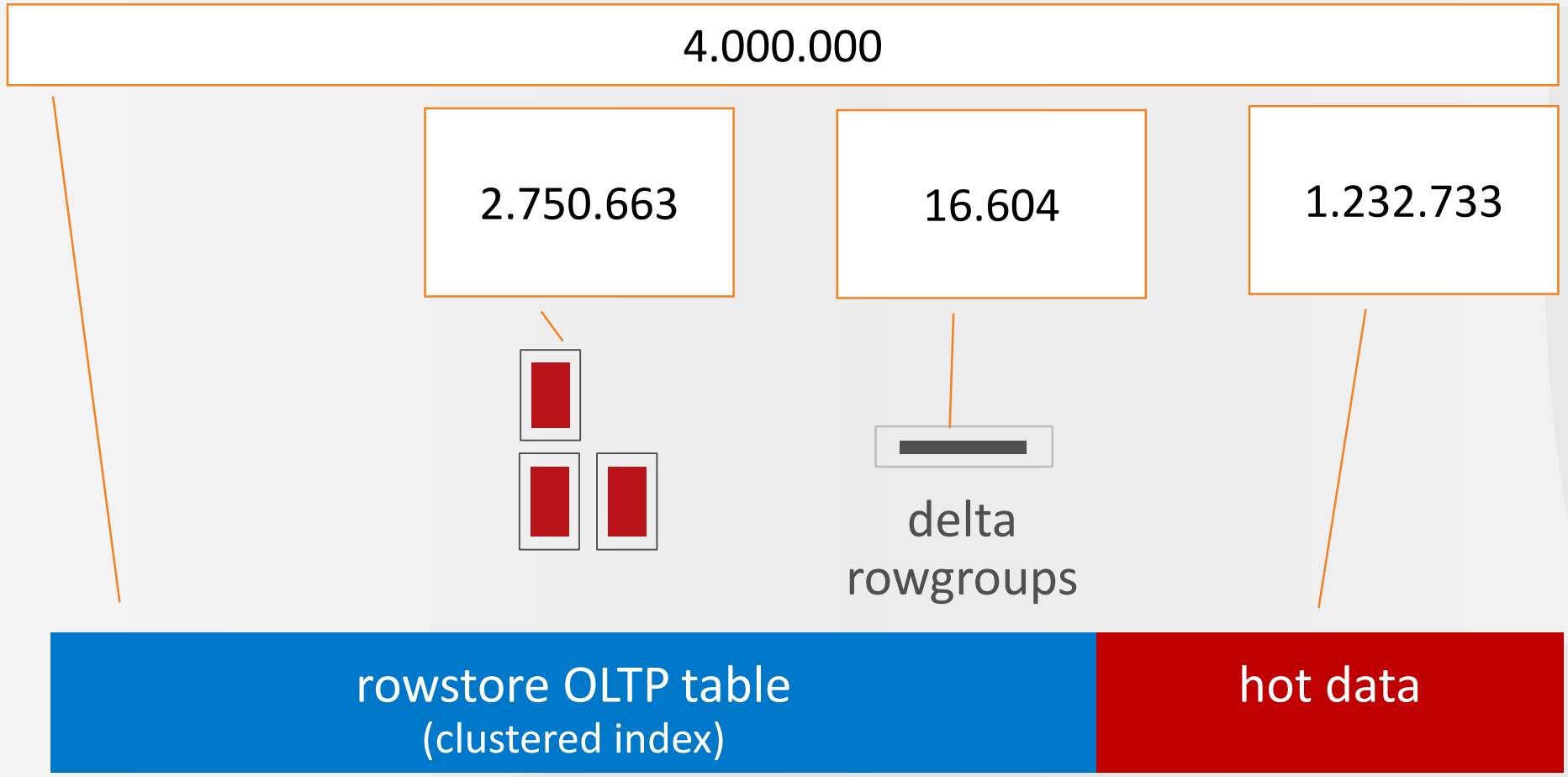
Demo

Scenario

- OLTP system: Order Management Application
- Table Order with account-key, customer name, order number, purchase price and status
- The table has 3 million records with order information

- Disk-based -> Create Nonclustered columnstore index
- In-memory -> Create Clustered columnstore index
- Load 1 million orders and these orders are not yet received by the customers.
- Run analytics queries

Real-time OLTP | Disk-Based



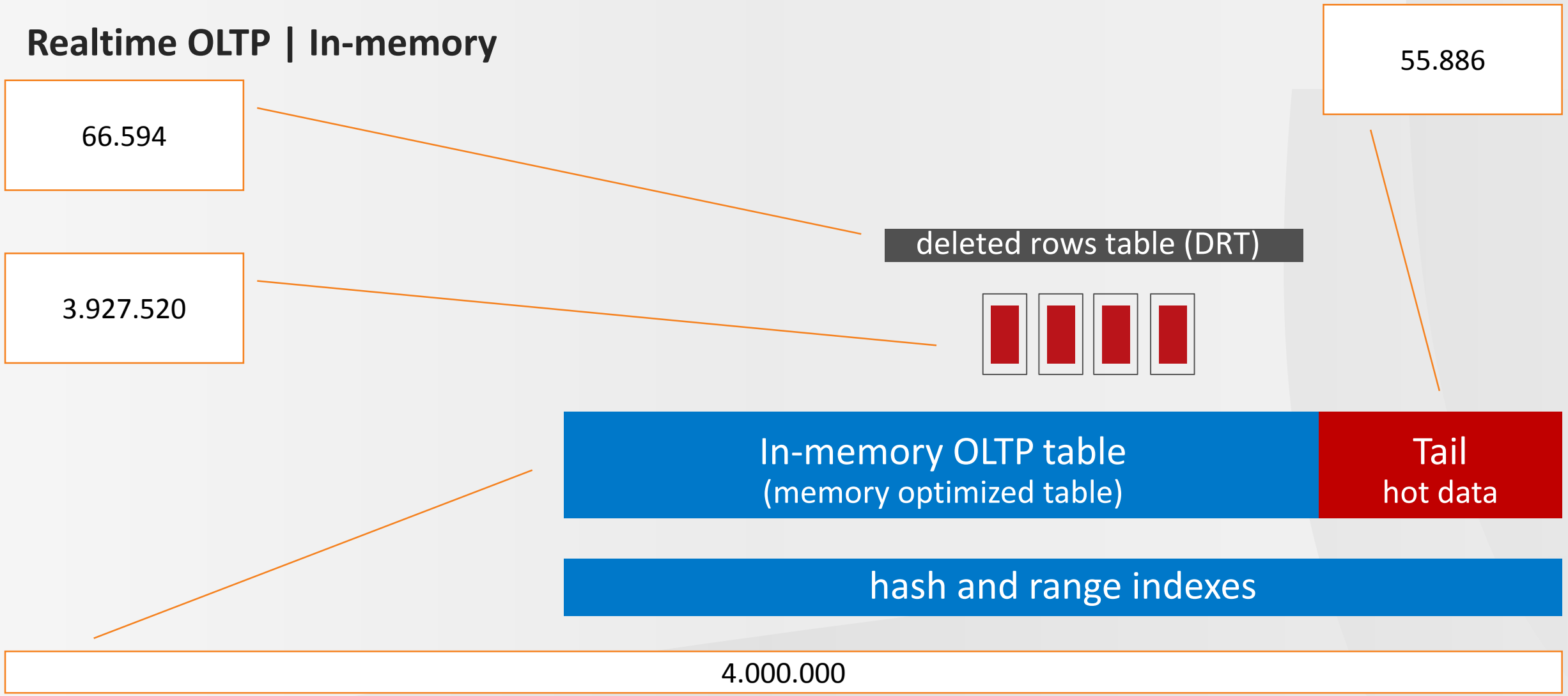
Real-time OLTP | Disk-Based

```
SUM(PurchasePrice),
MAX(PurchasePrice,
AVG(PurchasePrice)
```

	Rowstore	NCCI	NCCI with filter
Execute time	800 ms	200 ms	50 ms
CPU time	1000 ms	150 ms	10 ms
Logical reads	42.000	30.000	0

Segments: 4

Realtime OLTP | In-memory



Real-time OLTP | In-memory (CCI) versus disk Based (NCCI)

```
SUM(PurchasePrice),
MAX(PurchasePrice,
AVG(PurchasePrice)
```

	In-memory	Disk based
Execute time	51 ms	71 ms
CPU time	0 ms	16 ms
Segment reads	6	4

Data Solutions Trainingen

- SQL Server Performance Tuning and Optimization → 24 augustus en 11 oktober
- Updating Your Skills to SQL Server 2016 → 28 september

- The Essentials of Programming in R → 22 augustus
- Microsoft Azure Machine Learning → 08 september
- Microsoft Azure Data Solutions → 06 oktober
- Mastering and Optimizing DAX → 17 oktober

training.infosupport.com

Further Reading

Niko Neugebauer

SQL Server, Columnstore, Data Platform & Community

Kalen Delaney

In-Memory OLTP

InfoSupport Solid Innovator

Vragen?