Columnstore Indexes – Short Notes (SQL Server)

Source: https://sqlespresso.com/2019/06/26/ → **Author:** Monica Morehouse (Rathbun)

1. What Is a Columnstore Index?

- A columnstore index stores table data by **columns instead of rows**.
  In traditional rowstore (B-tree) indexes, all columns of a row are stored together.
- In a columnstore, **each column's data is stored separately**, which enables high compression and efficient scanning of large datasets.
- Data is divided into **rowgroups** (up to ~1 million rows each) and then into **column segments**. These segments are the basic storage unit for compression and query processing.
- There is also something called a **delta store**: a small buffer for new rows that haven't yet formed a full rowgroup. A background process called the **tuple-mover** moves completed rowgroups from the delta store into the compressed columnstore.
- **Batch mode execution** is used with columnstore indexes, processing many rows at once, which improves performance for analytical operations like aggregates.

2. Types of Columnstore Indexes

Clustered Columnstore Index (CCI)

- **Replaces the entire storage** of the table with a columnstore format.
- It compresses and stores all columns of the table, and is ideal for **large data warehouse fact tables**.
  https://medium.com/data-bloodies/understanding-the-columnstore-indexes-in-sql-server-f0e4fcaa52f3

Nonclustered Columnstore Index (NCCI)

- Added on top of an existing table that may already have a rowstore index.
- Useful when only certain columns benefit from analytics queries.

3. How Columnstore Improves Performance

Compression

- Columnstore formats typically achieve **much higher compression** than rowstore because similar data values in a column compress well.

Reduced I/O

- Because data is stored by column, SQL Server can **read only the columns needed by a query**, dramatically reducing disk I/O.

Batch Mode

- SQL Server processes many rows together in batch mode, providing significant performance improvements, especially for aggregates such as SUM(), AVG(), GROUP BY, etc.

Execution Plan Impact

- When a columnstore index is present, the optimizer often chooses **batch mode** and uses fewer operators with much lower cost than the equivalent rowstore plan, leading to big performance gains.

4. Creating a Columnstore Index

Clustered Example

```
CREATE CLUSTERED COLUMNSTORE INDEX CS_IDX_FactResellerSalesXL
ON dbo.FactResellerSalesXL;
```

- A table can only have **one clustered index** (whether rowstore or columnstore).

Nonclustered Example

- Create a columnstore index on specific columns if you want columnstore benefits without replacing the main storage format.

5. Limitations & Considerations

Not for OLTP

- Columnstore indexes are **designed for analytical workloads**, not high-transaction OLTP tables. Frequent inserts, updates, and deletes **reduce effectiveness**.

Table Size Requirements

- The technology benefits tables with **millions of rows** because small tables don't get effective segmentation and compression.

Data Volatility

- Tables with **high update/delete activity** can degrade compression and performance since modifications are expensive (handled as delete + insert under the hood).

Data Types

- Some data types aren't ideal or fully supported in older versions of SQL Server, though support has improved over time.

Not Ideal for Point Lookups

- Columnstore indexes excel for scans, aggregations, and analytics but are **not as good as rowstore indexes (B-trees) for single row lookups**.
  https://redmondmag.com/articles/2020/07/22/understanding-columnstore-indexes.aspx?

6. Best Use Cases

✅ Large data warehouse fact tables
✅ Tables primarily used for **analytics and reporting**
✅ Workloads with **aggregations and scans** over many rows
❌ Highly transactional OLTP tables
❌ Tables with frequent updates/deletes
❌ Queries requiring single row lookups or heavy seek patterns

Use columnstore for **read-intensive, large-scale analytics**, and combine with traditional rowstore indexes where needed for mixed workloads.

7. Performance Example Summary

- Running an aggregate query on an 11.6 million row table:
  - With rowstore: many logical reads and high CPU/time.
  - With columnstore: near-zero logical reads from rowstore, very fast batch mode execution, and **dramatic performance improvement**.

Source:

https://sqlespresso.com/2019/06/26/understanding-columnstore-indexes-in-sql-server-part-1/

https://sqlespresso.com/2019/07/10/understanding-columnstore-indexes-in-sql-server-part-2/

https://sqlespresso.com/2019/07/17/understanding-columnstore-indexes-in-sql-server-part-3/

Credits: Monica Morehouse (Rathbun)