**Columnstore Index:**

 It is a column-based index geared toward increasing query performance for workloads that involve large amounts of data, typically found in data warehouse fact tables.

**Table 2: Requirements and Limitations of Columnstore Index**

|  |  |
| --- | --- |
| Description | Requirement/Limitation |
| No. of columnstore indexes per table | 1 |
| Index record size limit of 900 bytes | No limit/Not applicable |
| Index limit of 16 key columns | No limit/Not applicable |
| Table partitioning support | Yes, as a partition aligned index. |
| Can be combined with row-based indexes? | Yes, if clustered index, all columns must be present in columnstore index. |
| Update, Delete, Insert, Merge supported? | No, columnstore indexes are read-only but workarounds exist. Refer to Books Online: Best Practices: Updating Data in a Columnstore Index. |

**Columnstore Index Architecture**

Physically, data is stored in rowgroups. Each rowgroup is subdivided into column segments, one per column in the table. The column segments are then compressed and stored on pages.

A picture containing text, building, clipart, window

Description automatically generated

Because columnstore indexes separate each column into its own compressed structure, any columns not needed for a query can be ignored.

Rowgroup structures can be viewed using the dynamic management view:

SELECT

tables.name AS table\_name,

indexes.name AS index\_name,

partitions.partition\_number,

column\_store\_row\_groups.row\_group\_id,

column\_store\_row\_groups.state\_description,

column\_store\_row\_groups.total\_rows,

column\_store\_row\_groups.size\_in\_bytes

FROM sys.column\_store\_row\_groups

INNER JOIN sys.indexes

ON indexes.index\_id = column\_store\_row\_groups.index\_id

AND indexes.object\_id = column\_store\_row\_groups.object\_id

INNER JOIN sys.tables

ON tables.object\_id = indexes.object\_id

INNER JOIN sys.partitions

ON partitions.partition\_number =

     column\_store\_row\_groups.partition\_number

AND partitions.index\_id = indexes.index\_id

AND partitions.object\_id = tables.object\_id

WHERE tables.name = ‘TechnicalResult'

ORDER BY tables.object\_id, indexes.index\_id,

column\_store\_row\_groups.row\_group\_id;

Similarly, we can view metadata about segments within each rowgroup using another dynamic management view: sys.column\_store\_segments.

SELECT

tables.name AS table\_name,

indexes.name AS index\_name,

columns.name AS column\_name,

partitions.partition\_number,

column\_store\_segments.segment\_id,

column\_store\_segments.row\_count,

column\_store\_segments.has\_nulls,

column\_store\_segments.on\_disk\_size

FROM sys.column\_store\_segments

INNER JOIN sys.partitions

ON column\_store\_segments.hobt\_id = partitions.hobt\_id

INNER JOIN sys.indexes

ON indexes.index\_id = partitions.index\_id

AND indexes.object\_id = partitions.object\_id

INNER JOIN sys.tables

ON tables.object\_id = indexes.object\_id

INNER JOIN sys.columns

ON tables.object\_id = columns.object\_id

AND column\_store\_segments.column\_id = columns.column\_id

WHERE tables.name = ‘TechnicalResult'

This query returns a row per segment, allowing you to analyze the contents of a columnstore index for each column. This can help determine which columns compress most (or least) effectively, as well as keep track of the size and structure of a columnstore index on a more granular scale

Server Name: **UKDVDB159**

SELECT \* FROM TechnicalHub.fct.TechnicalResult WHERE FK\_DataSet = 6

**Without Columnstore Index**:

DROP INDEX [CIX\_FCT\_TechnicalResult] ON [fct].[TechnicalResult]

SQL Server parse and compile time: 10:40 Mins

CPU time = 0 ms, elapsed time = 7 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 0 ms.

(24250371 rows affected)

Table 'TechnicalResult'. Scan count 1, logical reads 4353565, physical reads 0, read-ahead reads 4352098, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:

CPU time = 83219 ms, elapsed time = 640774 ms.

Completion time: 2021-12-10T20:04:38.5400566+05:30

**With Columnstore Index:** 11:44 Mins

CREATE CLUSTERED

COLUMNSTORE INDEX [CIX\_FCT\_TechnicalResult] ON [fct].[TechnicalResult] WITH (DROP\_EXISTING = OFF, COMPRESSION\_DELAY = 0) ON [PRIMARY]

GO

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 0 ms.

Table 'TechnicalResult'. Scan count 6, logical reads 4425728, physical reads 1, read-ahead reads 4847012, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 64615, lob physical reads 21, lob read-ahead reads 172700.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 3653, lob physical reads 14, lob read-ahead reads 3661.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 9325, lob physical reads 15, lob read-ahead reads 15525.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 19421, lob physical reads 15, lob read-ahead reads 40241.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 1558, lob physical reads 6, lob read-ahead reads 0.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 44047, lob physical reads 10, lob read-ahead reads 108573.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

Table 'TechnicalResult'. Scan count 4, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 1540, lob physical reads 6, lob read-ahead reads 0.

Table 'TechnicalResult'. Segment reads 299, segment skipped 0.

SQL Server Execution Times:

CPU time = 1612688 ms, elapsed time = 704788 ms.

Completion time: 2021-12-10T19:36:26.5198528+05:30