Compression

Wednesday, September 15, 2021 2:24 PM

Row compression helps store data more efficiently in a row by storing fixed-length data types in variable-length storage format. A compressed row uses 4 bits per compressed column to store the length of the data in the column. NULL and 0 values across all data types take no additional space other than these 4 bits.
Page compression is a superset of row compression. In addition to storing data efficiently inside a row, page compression optimizes storage of multiple rows in a page, by minimizing the data redundancy. Page compression uses prefix compression and dictionary compression. Prefix compression looks for common patterns in the beginning of the column values on a given column across all rows on each page. Dictionary compression looks for exact value matches across all columns and rows on each page.

Some examples of the flexibility in applying data compression are to:

- Row-compress some tables, page-compress some others, and don't compress the rest.
- Page-compress a heap or clustered index, but have no compression on its non-clustered indexes.
- Row-compress one index, and have no compression on another index.
- Row-compress some partitions of a table, page-compress some others, and don't compress the rest.

From <https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008/dd894051(v=sql.100)?redirectedfrom=MSDN>

1. **Assess Largest tables:**

```
SELECT
   t.NAME AS TableName,
   s.Name AS SchemaName,
        t.create_date Created,
   p.rows AS RowCounts,
   SUM(a.total_pages) * 8 AS TotalSpaceKB,
   SUM(a.used_pages) * 8 AS UsedSpaceKB,
   (SUM(a.total_pages) - SUM(a.used_pages)) * 8 AS UnusedSpaceMB
FROM
   sys.tables t
INNER JOIN
   sys.indexes i ON t.OBJECT_ID = i.object_id
INNER JOIN
   sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
INNER JOIN
   sys.allocation_units a ON p.partition_id = a.container_id
LEFT OUTER JOIN
   sys.schemas s ON t.schema_id = s.schema_id
   --WHERE
   ---- t.NAME NOT LIKE 'dt%'
         --t.NAME Like '%BU%'
         --or t.NAME Like '%BAK%'
   --AND t.is_ms_shipped = 0
   --AND i.OBJECT_ID > 255
GROUP BY
   t.Name, s.Name, p.Rows, t.create_date
ORDER BY
   SUM(a.total_pages) desc
```

2. Check the effect of Compression:

```
EXEC sp_estimate_data_compression_savings 'Production', 'WorkOrderRouting', NULL,
NULL, 'ROW' ;
```

a. Parameters to Assess:
NONE, ROW, PAGE, COLUMNSTORE, or COLUMNSTORE_ARCHIVE

sp_estimate_data_compression_savings (Transact-SQL) - SQL Server | Microsoft Docs

3. Check the % of UPDATES and SCANS on the target table

U and S measurements for the largest tables in the database, and targeted tables with S greater than 75 percent, U less than 20 percent for page compression.

U: Percent of Update Operations on the Object

```
SELECT o.name AS [Table_Name], x.name AS [Index_Name],
     i.partition_number AS [Partition],
     i.index_id AS [Index_ID], x.type_desc AS [Index_Type],
     i.range_scan_count * 100.0 /
        (i.range_scan_count + i.leaf_insert_count
         + i.leaf_delete_count + i.leaf_update_count
         + i.leaf_page_merge_count + i.singleton_lookup_count
        ) AS [Percent_Scan]
FROM sys.dm_db_index_operational_stats (db_id(), NULL, NULL, NULL) i
JOIN sys.objects o ON o.object_id = i.object_id
JOIN sys.indexes x ON x.object_id = i.object_id AND x.index_id = i.index_id
WHERE (i.range_scan_count + i.leaf_insert_count
     + i.leaf_delete_count + leaf_update_count
     + i.leaf_page_merge_count + i.singleton_lookup_count) != 0
AND objectproperty(i.object_id,'IsUserTable') = 1
ORDER BY [Percent_Scan] DESC
```

S: Percent of Scan Operations on the Object

```
SELECT o.name AS [Table_Name], x.name AS [Index_Name],
     i.partition_number AS [Partition],
     i.index_id AS [Index_ID], x.type_desc AS [Index_Type],
     i.range_scan_count * 100.0 /
        (i.range_scan_count + i.leaf_insert_count
         + i.leaf_delete_count + i.leaf_update_count
         + i.leaf_page_merge_count + i.singleton_lookup_count
        ) AS [Percent_Scan]
FROM sys.dm_db_index_operational_stats (db_id(), NULL, NULL, NULL) i
JOIN sys.objects o ON o.object_id = i.object_id
JOIN sys.indexes x ON x.object_id = i.object_id AND x.index_id = i.index_id
WHERE (i.range_scan_count + i.leaf_insert_count
     + i.leaf_delete_count + leaf_update_count
     + i.leaf_page_merge_count + i.singleton_lookup_count) != 0
AND objectproperty(i.object_id,'IsUserTable') = 1
ORDER BY [Percent_Scan] DESC
```

Example

Here is an example how a customer used these measurements to decide which tables to page-compress. The customer had an OLTP database running on a server with average CPU utilization of approximately 20 percent. The large amount of available CPU, the significant amount of planned database growth, and the expense of storage provided motivation for data compression. The customer computed space savings, and the U and S measurements for the largest tables in the database, and targeted tables with S greater than 75 percent, U less than 20 percent for page compression. Table 1 shows the estimated row and page savings, the values of S and U, and the decision as to whether to row or page compress.

Table	Savings ROW %	Savings PAGE %	S	U	Decision	Notes
T1	80%	90%	3.80%	57.27%	ROW	Low S, very high U. ROW savings close to PAGE
T2	15%	89%	92.46%	0%	PAGE	Very high S
T3	30%	81%	27.14%	4.17%	ROW	Low S
T4	38%	83%	89.16%	10.54%	ROW	High U
T5	21%	87%	0.00%	0%	PAGE	Append ONLY table
T6	28%	87%	87.54%	0%	PAGE	High S, low U
T7	29%	88%	0.50%	0%	PAGE	99% appends
T8	30%	90%	11.44%	0.06%	PAGE	85% appends
T9	84%	92%	0.02%	0.00%	ROW	ROW savings ~= PAGE
T10	15%	89%	100.00%	0.00%	PAGE	Read ONLY table

[Data Compression: Strategy, Capacity Planning and Best Practices | Microsoft Docs]