# Testing Columnstore Row Groups Trimming

## What is Row Groups Trimming?

It's basically an event that happens when a Row Group doesn't reach the maximum allowed number of rows (1 048 576) but getting closed and compressed before.

## Documented Row Group Trim Reasons

According to the Microsoft documentation, we can have the following Trim Reasons:

0) **UNKNOWN_UPGRADED_FROM_PREVIOUS_VERSION**: Occurred when upgrading from the previous version of SQL Server.
1) **NO_TRIM**: The row group was not trimmed. The row group was compressed with the maximum of 1,048,476 rows. The number of rows could be less if a subset of rows was deleted after delta rowgroup was closed.
2) **BULKLOAD**: The bulk load batch size limited the number of rows.
3) **REORG**: Forced compression as part of REORG command.
4) **DICTIONARY_SIZE**: Dictionary size grew too big to compress all of the rows together.
5) **MEMORY_LIMITATION**: Not enough available memory to compress all the rows together.
6) **RESIDUAL_ROW_GROUP**: Closed as part of last row group with rows < 1 million during index build operation.
7) **STATS_MISMATCH**: Only for columnstore on in-memory table. If stats incorrectly indicated >= 1 million qualified rows in the tail but we found fewer, the compressed rowgroup will have < 1 million rows.
8) **SPILLOVER**: Only for columnstore on in-memory table. If tail has > 1 million qualified rows, the last batch remaining rows are compressed if the count is between 100k and 1 million.

We are going to test some case scenarios of Row Group Trimming.

# Test Case Scenario 1 – NO_TRIM reason

## Test Case Description:

The row group was not trimmed. The row group was compressed with the maximum of 1,048,476 rows. The number of rows could be less if a subset of rows was deleted after delta rowgroup was closed.

## Steps taken to test the NO_TRIM reason:

1.) Bulk load 5 batches of size 1,048,476 and review the row groups physical stats: **trim_reason_desc** column:

```sql
SELECT
    OBJECT_NAME(object_id) AS table_name,
    row_group_id,
    total_rows,
    state_desc,
    trim_reason,
    trim_reason_desc
FROM
    sys.dm_db_column_store_row_group_physical_stats
WHERE
    object_id = OBJECT_ID('Production.TransactionHistory_DST')
ORDER BY
    row_group_id ASC;
```

100 %

Results | Messages

| | table_name | row_group_id | total_rows | state_desc | trim_reason | trim_reason_desc |
|---|---|---|---|---|---|---|
| 1 | TransactionHistory_DST | 0 | 1048576 | COMPRESSED | 1 | NO_TRIM |
| 2 | TransactionHistory_DST | 1 | 1048576 | COMPRESSED | 1 | NO_TRIM |
| 3 | TransactionHistory_DST | 2 | 1048576 | COMPRESSED | 1 | NO_TRIM |
| 4 | TransactionHistory_DST | 3 | 1048576 | COMPRESSED | 1 | NO_TRIM |
| 5 | TransactionHistory_DST | 4 | 1048576 | COMPRESSED | 1 | NO_TRIM |

As you can see, the **trim_reason_desc** column has only **NO_TRIM** values as we used the maximum RG size: 1 048 576 rows.

# Test Case Scenario 2 – BULKLOAD trim reason

## Test Case Description:

Load 1 batch of size 500 000, so we will get 1 compressed rowgroups with BULKLOAD trim reason.

## Steps taken to test the BULKLOAD trim reason:

1.) Bulk load 1 batch of size 500 000 rows and review the row groups physical stats: **trim_reason_desc** column:

```
SELECT
    OBJECT_NAME(object_id) AS table_name,
    row_group_id,
    total_rows,
    state_desc,
    trim_reason,
    trim_reason_desc
FROM
    sys.dm_db_column_store_row_group_physical_stats
WHERE
    object_id = OBJECT_ID('Production.TransactionHistory_DST')
ORDER BY
    row_group_id ASC;
```

100 %

Results | Messages

| | table_name | row_group_id | total_rows | state_desc | trim_reason | trim_reason_desc |
|---|---|---|---|---|---|---|
| 1 | TransactionHistory_DST | 0 | 500000 | COMPRESSED | 2 | BULKLOAD |

As you can see, the **trim_reason_desc** column has only BULKLOAD values as we did not use the maximum RG size: 1 048 576 rows, but lower.

# Test Case Scenario 3 – DICTIONARY_SIZE trim reason

## Test Case Description:

DICTIONARY_SIZE: Dictionary size grew too big to compress all of the rows together. Load 1 batch of size 500 000, one of the columns should contain a long VARCHAR value that exceeds the dictionary size (16 MB). We will get multiple rowgroups with the DICTIONARY_SIZE trim reason.

## Steps taken to test the DICTIONARY_SIZE trim reason:

1.) Bulk load 1 batch of size 500 000 rows (one of the columns on the CCI should be a large varchar column) and review the row groups physical stats: **trim_reason_desc** column:

```sql
SELECT
    OBJECT_NAME(object_id) AS table_name,
    row_group_id,
    total_rows,
    state_desc,
    trim_reason,
    trim_reason_desc
FROM
    sys.dm_db_column_store_row_group_physical_stats
WHERE
    object_id = OBJECT_ID('Production.TransactionHistory_DST')
ORDER BY
    row_group_id ASC;
```

100 %

Results | Messages

| | table_name | row_group_id | total_rows | state_desc | trim_reason | trim_reason_desc |
|---|---|---|---|---|---|---|
| 1 | TransactionHistory_DST | 0 | 137561 | COMPRESSED | 4 | DICTIONARY_SIZE |
| 2 | TransactionHistory_DST | 1 | 137561 | COMPRESSED | 4 | DICTIONARY_SIZE |
| 3 | TransactionHistory_DST | 2 | 137561 | COMPRESSED | 4 | DICTIONARY_SIZE |
| 4 | TransactionHistory_DST | 3 | 87317 | OPEN | NULL | NULL |

As we can see, we did not get one compressed RG, but multiple smaller row groups with the trim reason DICTIONARY_SIZE (as we exceeded the dictionary limit 16 MB).