Generic Columnstore Migration Documentation

- Author: Ivan Radchenko
- Created: 28/11/2021
- Version: 1.0.0

Thank you for using my solution! If you have any questions, feel free to e-mail me at lardo.ag@gmail.com.

Table of Contents

---

1. Overview

This is a step-by-step guide that will help you migrate your table to Columnstore with ease. If you're not sure which tables are the best candidates for Columnstore, you might want to use Niko's sript to determine it. Niko also offers a simple migration solution, that drops all indexes on a given table and creates a Columnstore index instead. If your table is small enough, you can afford the downtime and you're sure that you won't need to roll back the migration, go ahead and use his solution. However, if you find this approach too heavy, risky, or whatever else that prevents you from using it, that's where I come in. My solution splits the migration into the following steps:

1. Create a new partition function and partition scheme for your table. This is totally optional, since you can leave your table non-partitioned or use an existing partition scheme you might have.
2. Create the target Columnstore table side-by-side.
3. Load the target table, all at once or incrementally, depending on various considerations.
4. Create and update statistics on the target table.
5. Align source and target data, so that the tables are identical and ready for the switch.
6. Switch the tables, by renaming the source table and then assigining the source table name to the target. Only the last two steps require downtime, which is as short as I could make it. Should the need arize, rollback will also be relativetively simple, as it only involves last two steps as well. Each steps contains a .sql script, that can either execute the step directly, or print the script for this change. It's highly recommended not to execute the change directly in a production environment. Instead, it's preferable to execute it in a testing environment, printing out the final scripts. After you have all the scripts, verify that they're working correctly, make necessary adjustments, and only then execute them in production. In case the scripts neeeded manual

adjustments or didn't work as intended, please notify me at <ivanr@madeiradata> and I'll do my best to fix this in a new version.

---

2. Creating a new partition function and partition scheme

The following script generates a command that creates a dedicated partition function and partition scheme (PF&PS) for a desired table. The script accepts the following parameters:

- @DBName sysname = 'MyDB' - database, where the target table is located.
- @SchemaName sysname = 'MyTargetSchema' - schema, where the target table is located.
- @TableName sysname = 'MyTargetTable' - target table name.
- @PartitionFunctionType nvarchar(50) = 'datatype' - desired datatype of partitioning column (e.g. 'datetime', 'datetime2', etc...). Currently only date types are supported.
- @PartitionFunctionRange nvarchar(10) = 'RIGHT' - desired PF range ('LEFT' or 'RIGHT'). If unsure, leave 'RIGHT' as default.
- @StartTime datetime = 'yyyymmdd' - date that corresponds to the first partition
- @EndTime datetime = 'yyyymmdd' - date that corresponds to the last partition
- @FG nvarchar(50) = 'PRIMARY' - desired filegroup for the table to reside on. If unsure, leave 'PRIMARY' as default.
- @ExecuteCommands BIT = 0 - execute the PF&PS creation (1) or print the script (0).

---

3. Creating the target Columnstore table

The following script generates a command that creates the target Columnstore table. First of all, it requires sp_GenerateTableDDLScript (https://github.com/EitanBlumin/sp_GenerateTableDDLScript) to exist in [master] DB. The script accepts the following parameters:

- @SourceDBName = 'MySourceDB' - database, where the source table is located.
- @SourceSchemaName sysname = 'MySourceSchema' - schema, where the source table is located.
- @SourceTableName sysname = 'MySourceTable' - source table name.
- @TargetDBName sysname = 'MyTargetDB' - database, where the target table is located.
- @TargetSchemaName sysname = 'MyTargetSchema' - schema, where the target table is located.
- @TargetTableName sysname = 'MyTargetTable' - target table name.
- @TargetFG sysname = NULL - desired filegroup for the table to reside on. Specify this if you want the table to be non-partitioned, otherwise leave NULL. Choose either @TargetFG or @TargetPartitionScheme.
- @TargetPartitionScheme sysname = NULL - desired PS for the table to reside on. Specify this if you want the table to be partitioned, otherwise leave NULL.
- @TargetPartitionColumn sysname = NULL--'msgCreateDate' - desired column to use for partitioning. Specify this in conjunction with @TargetPartitionScheme, otherwise leave NULL.
- @CompressionMode NVARCHAR(20) = 'COLUMNSTORE_ARCHIVE' - desired table compression mode. Can be either 'COLUMNSTORE' or 'COLUMNSTORE_ARCHIVE', there's

a tradeoff between CPU and I/O to be considered. The former offers weaker compression and is less CPU-intensive, while latter is the opposite.
- @IncludeIdentity bit = 1 - specify if the target table includes an identity column (1) or not (0)
- @DropTargetTable bit = 1 - target table will be dropped and recreated if exists (1) or left as is (0).
- @ExecuteCreation bit = 0 - execute the table creation (1) or print the script (0).

---

4. Loading the target Columnstore table

The following script generates a command that migrates data from the source to the target table. The script accepts the following parameters:

- @SourceDBName sysname = 'MySourceDB' - database, where the source table is located.
- @SourceSchemaName sysname = 'MySourceSchema' - schema, where the source table is located.
- @SourceTableName sysname = 'MySourceTable' - source table name.
- @TargetDBName sysname = 'MyTargetDB' - database, where the target table is located.
- @TargetSchemaName sysname = 'MyTargetSchema' - schema, where the target table is located.
- @TargetTableName sysname = 'MyTargetTable' - target table name.
- @IncrementColumn sysname = NULL - desired source table column to determine the increment. Choose @IncrementColumn according to source partition/CI column to avoid full table scans on every increment. If you want to load the whole table at once, leave NULL.
- @IncrementValue varchar(10) = NULL - specify this in conjunction with @IncrementColumn, if you want to load the target table incrementally:
  - if your @IncrementColumn is a date, define @IncrementValue as 'YY, 1'/'MM, 1'/'DD, 1', depending on the desired chunk size.
  - if your @IncrementColumn is a number, define @IncrementValue as '1045678' or more, depending on the desired chunk size.
- @EndDate nvarchar(8) = CONVERT(nvarchar(8), GETDATE(), 112) -- desired last day in the target table.
- @IncludeIdentity bit = 1 - specify if the target table includes an identity column (1) or not (0).
- @ExecuteInsert bit = 0 - execute the table load process (1) or print the script (0).

---

5. Creating statistics

Creates statictics on all columns and updates them. Requires Auto Create Statistics on the target DB. The script accepts the following parameters:

- @SourceDBName sysname = 'MySourceDB' - database, where the source table is located.
- @SourceSchemaName sysname = 'MySourceSchema' - schema, where the source table is located.
- @SourceTableName sysname = 'MySourceTable' - source table name.

- @TargetDBName sysname = 'MyTargetDB' - database, where the target table is located.
- @TargetSchemaName sysname = 'MyTargetSchema' - schema, where the target table is located.
- @TargetTableName sysname = 'MyTargetTable' - target table name.
- @TargetPartitionColumn sysname = NULL - specify the partitioning column. If the table is non-partitioned, leave NULL.
- @StartDate datetime = CAST(GETDATE()-4 AS DATE) - specify in conjunction with @TargetPartitionColumn. Four days worth of data should be enough (default), if the stats weren't created, increase it.
- @TopRows int = 10000 - 10000 (default) should be enough, if the stats weren't created, increase it
- @ExecuteCommands bit = 1 - execute the statistic creation process (1) or print the script (0).

---

6. Aligning source and target data

The following script generates the DELETE/INSERT commands that load latest data (modified since the main migration) from source to target table. The script accepts the following parameters:

- @SourceDBName sysname = 'MySourceDB' - database, where the source table is located.
- @SourceSchemaName sysname = 'MySourceSchema' - schema, where the source table is located.
- @SourceTableName sysname = 'MySourceTable' - source table name.
- @TargetDBName sysname = 'MyTargetDB' - database, where the target table is located.
- @TargetSchemaName sysname = 'MyTargetSchema' - schema, where the target table is located.
- @TargetTableName sysname = 'MyTargetTable' - target table name.
- @TargetPKColumns varchar(MAX) = 'PKcolumn1,PKcolumn2,...,PKcolumnN' - In DELETE/INSERT commands, PK columns should be used as the JOIN predicate. If there's no PK, choose a set of columns that uniquely identify a row and are rarely updated.
- @TargetPartitionColumn sysname = NULL - specify the partitioning column. If the table is non-partitioned, leave NULL.
- @TargetUpdateIndicator sysname = choose a target table column to indicate when the row was last updated, to avoid synching the whole table.
- @StartDate datetime = CAST(GETDATE()-1 AS date) -- set it one day before the main migration (script #2) start date, specify in conjunction with @TargetUpdateIndicator. If you want to sync the whole table, leave NULL.
- @IncludeIdentity bit = 1 - specify if the target table includes an identity column (1) or not (0)
- @ExecuteCommands bit = 0 - execute the table sync process (1) or print the script (0).

---

7. Switching tables

The following script switches target table to be the live production table, by renaming the source and the target tables accordingly. The script accepts the following parameters:

- @SourceDBName sysname = 'MySourceDB' - database, where the source table is located.
- @SourceSchemaName sysname = 'MySourceSchema' - schema, where the source table is located.
- @SourceTableName sysname = 'MySourceTable' - source table name.
- @SourceTableSuffix nvarchar(10) = 'MySuffix' - desired suffix for the source table after it's renamed.
- @TargetDBName sysname = 'MyTargetDB' - database, where the target table is located.
- @TargetSchemaName sysname = 'MyTargetSchema' - schema, where the target table is located.
- @TargetTableName sysname = 'MyTargetTable' - target table name.
- @ExecuteCommands bit = 0 - execute the table switch process (1) or print the script (0).