Using SQL Server 2012 Column-Store with SAP BW

Applies to:

SAP Business Warehouse 7.0 and higher running on Microsoft SQL Server 2012 and higher.

Summary

SQL Server 2012 introduced an additional storage format for data. Besides the normal row-oriented store, there is a column-oriented store as well. To make it as transparent as possible for database administrators and applications, the new column-store was implemented as a new index type: non-clustered column-store index. Therefore, a single table can use both stores side by side.

The column-oriented store is optimized for aggregating mass data, while the row-oriented store can be used optimal when accessing just a single row. SQL Server 2012 column-store support has been integrated into SAP BW and was down-ported to all common SAP BW releases:

- SAP BW 7.0 (SAP NetWeaver 7.0)
- SAP BW 7.01 (SAP enhancement package 1 for SAP NetWeaver 7.0)
- SAP BW 7.02 (SAP enhancement package 2 for SAP NetWeaver 7.0)
- SAP BW 7.11 (SAP enhancement package 1 for SAP NetWeaver 7.1)
- SAP BW 7.3 (SAP NetWeaver 7.3)
- SAP BW 7.31 (SAP enhancement package 1 for SAP NetWeaver 7.3)

In all these SAP BW releases you can benefit from increased query performance and reduced disk space requirements of the column-store.

Author: Martin Merdes

Company: Microsoft

Created on: October 2012

Updated on: December 2012

Author Bio

Martin Merdes is a Senior Software Development Engineer at Microsoft. Since 1996 he has been working in different technical functions for Microsoft at SAP Germany: Support Engineer, Senior Consultant, Technical Account Manager, Premier Field Engineer and Software Development Engineer. Since 2009 he is responsible for the porting of SAP BW on Microsoft SQL Server.

Table of Contents

Column-Store in SQL Server 2012

The SQL Server 2012 introduces a new data warehouse query acceleration feature based on a new type of index called an xVelocity memory optimized column-store. This new index, combined with enhanced query processing features, improves data warehouse query performance by factors. For an introduction of column-store indexes, see http://msdn.microsoft.com/en-us/library/gg492088.aspx.

Besides improved query performance, you can further benefit from using the column-store:

Space Saving

Indexes can be compressed much better when stored by columns rather than by rows. The compression ratio depends on the actual data. Typically column-store indexes are by factors smaller than b-tree indexes. For columns with low cardinality, the compression can be hundreds times better and more.

There is no column order in column-store indexes. For b-tree indexes, it makes a difference whether you have an index on columns (A,B,C) or on columns (C,B,A), when using a filter on column A within a query. This is not the case for the column-store. Therefore, you can drop all secondary b-tree indexes once you have created a column-store index on all columns. This results in further space savings.

Query Performance

SQL Server 2012 leverages several optimizations to speed-up query performance of column-store indexes compared with b-trees:

- Index compression
  As described above, column-store indexes are much smaller than b-trees. The reduced index size results in faster index scanning.

- Parallelism
  The query algorithms for the column-store are optimized for parallel execution (batch mode). You can observe best query performance, if there are always at least two CPU threads available to run the query. Otherwise SQL Server is running the query without parallelism (row mode).

- Optimized memory structures
  The column-store is not using fixed sized SQL Server data pages from the buffer pool. Instead the column-store has its own memory pool. However, you do not need to configure this pool. Everything is done automatically by SQL Server memory management. You only have to make sure that sufficient memory is available for SQL Server.

Fast Index creation

Index creation can be parallelized much butter for column-store indexes than for b-trees. Indeed, index creation scales well with millions of rows and dozens of CPU threads. However, you will not benefit from parallelism during index creation for tables having less than a few million rows.

SQL Server restrictions

SQL Server Column-Store is only available in the Enterprise Edition of SQL Server 2012. For SAP BW this is not a limitation, because the Enterprise Edition is required anyway.

The non-clustered column-store in SQL Server 2012 is read-only. Therefore you cannot modify the data of a table as long as a column-store index exists for this table. Before loading data into the table, you first have to drop the column-store index. After the data load you should re-create the column-store index to optimize the query performance again. In SAP BW, these steps are performed automatically during BW cube compression.

The column-store supports the data type DECIMAL up to a precision of 18 digits, which is fully sufficient for a standard SAP BW. However, there are a few SAP solutions based on SAP BW that require a precision of more than 18 digits.

For using the column-store with SAP BW, you need at least cumulative update 2 for SQL Server 2012. However, we recommend installing service pack 1 instead (which contains all fixes from CU2). You can download the service pack from http://www.microsoft.com/en-us/download/details.aspx?id=35575.

Column-Store Support in SAP BW

SQL Server 2012 Column-Store support has been back-ported to SAP BW 7.0. There is no need to upgrade the SAP BW system to a newer release. Not even is it necessary to apply an enhancement package (7.01, 7.02 etc.). However, you have to install a minimum support package stack in order to benefit from the column-store.

SAP BW	Support Package Stack (SPS)	Release to Customers (RTC)
7.0	27 (SAP_BASIS SP27, SAP_BW SP29)	2012-07-26
7.01	12	2012-08-13
7.02	12	2012-09-03
7.11	*10*	2012-08-14 ***not yet supported for SQL Server 2012***
7.3	8	2012-10-02
7.31	5	2012-10-22

Note: SQL Server 2012 is not yet validated by SAP for SAP NetWeaver 7.11. Once SQL Server 2012 is supported for NetWeaver 7.11, you will find more information about the minimum SAP support package in SAP note 1651862.

Prerequisites for SAP BW

1. You first have to apply the minimum Support Package Stack as listed in the table above.

2. After that, you have to import the correction instruction of SAP note 1771177 using transaction SNOTE. The correction instruction contains report MSSCSTORE (see below). It further contains several improvements, which are not contained in the support packages mentioned above.

3. Finally, apply service pack 1 of SQL Server 2012 (or any newer service pack of SQL Server 2012)

Note: When applying the correction instruction of SAP note 1771177, you will get the message “Object can only be created in SAP package” (for report MSSCSTORE). Confirm this message by pressing the *OK* button.

SAP note 1771177 always contains the newest correction instructions that are required for the column-store. Therefore, you should apply this note even if you are running a newer SAP support package stack as listed above.

Usage of Column-Store

In SAP BW each cube has two fact tables. The f-fact table is optimized for data load, while the e-fact table is optimized for query performance. SQL Server 2012 Column-Store can be leveraged for SAP BW e-fact tables. A typical scenario looks like this:

- Data is loaded several times a day into SAP BW. Using a BW data transfer process, new data is inserted into the f-fact table of a cube. A request ID is added to each data row to identify the data transfer process that inserted the row. The f-fact table has the classical index layout with a separate row-store index (b-tree) per dimension. This results in typically 10 to 16 indexes per f-fact table.
- Once a day (or less frequently) a BW cube compression runs, typically at night. The BW cube compression moves (and aggregates) data from the f-fact to the e-fact table. The e-fact table has a single (unique, clustered) b-tree index: the P-index. In addition, there is one column-store index on all columns of the e-fact table: the CS-index. The total size of the cube is reduced by 3 factors:
    - The aggregation of the data (by removing the request ID)
    - The reduction of the number of indexes from typically 10 to 2
    - The extreme good compression skills of the column-store

  Customers may compress all requests of the f-fact table or compress only older requests (for example requests older than 7 days). As a result, all data (or at least most of the data) of the cube is stored in the e-fact table.
- During the cube compression, the CS-index is dropped and re-created. Therefore, it does not matter, that the column-store is read-only. The index creation for a column-store index is much faster than for a b-tree index. It scales quite well with the number of CPU threads. By default, SAP BW requests eight database threads for index creation.
- An SAP BW query is broken down into two database queries: One is running on the f-fact table, the other is running in parallel on the e-fact table. The database query on the e-fact table can fully benefit from the column-store. The database query on the f-fact table is also fast, since the f-fact table only contains a small portion of the cube data.

SAP BW restrictions

There are a few restrictions when using the column-store with SAP BW. For the latest information, see SAP note 1771177.

Key figures with increased precision

SQL Server 2012 column-store does not support the data type DECIMAL, if the precision is higher than 18 digits. The data type DECIMAL is used in SAP BW for key figures. In SAP BW you cannot create a key figure with a higher precision than 17 digits on your own (using SAP transaction RSD1). However, SAP can deliver such key figures using BW Content. There are a few SAP applications based on SAP BW (for example SAP BPC) that use such key figures with increased precision. Therefore, SAP BW simply does not use the column-store for cubes with increased precision. This is independent from the column-store configuration defined in report MSSCSTORE (see below).

SAP system upgrade to 7.3

Any SAP system upgrade of BW systems using the column-store works fine. However, when upgrading from 7.0x to 7.3x, the column-store indexes may be dropped during the SAP upgrade. They have to be re-created after the SAP system upgrade using report MSSCSTORE. This issue will be fixed in SAP Software Upgrade Manager (SUM) in the next support package (SL Toolset 1.0 support package 6).

BW Transport Connection

The column-store property of a BW cube is locally defined. It will not be transported from a BW test system to a BW productive system. When a cube is transported, it is created using the system default (either with or without column-store) on the target system.
SAP System Migration and System Copy (using either DB restore or R3load) are fully aware of the column-store. When using R3load, do not forget to run report SMIGR_CREATE_DDL on the source system and report RS_BW_POST_MIGRATION on the target system. For details, check the SAP System Copy Guide.

SAP Data Dictionary Support

Generally, the SAP Data Dictionary (DDIC) only supports indexes with up to 16 columns. Furthermore, the DDIC does not support special index types, like column-store. Therefore, the SAP DDIC was extended by table MSSSTORAGE, which persistently stores SQL Server specific database storage parameters. MSSSTORAGE contains the column-store configuration per table, which defines whether or not a column-store index is used. In addition, MSSTORAGE contains a system default, which is used for new tables.

The column-store configuration in SAP DDIC is taken into account for the following activities:

- Cube activation in SAP BW
- Index checking and creation in SAP BW
- Index consistency checks in SAP DDIC
- Table conversion in SAP DDIC (for example in SAP transaction SE14)
- SAP system copy using R3load (if SMIGR_CREATE_DDL is properly executed)

When transporting (for example from a development to a productive system), a cube is always created using the system default on the target system. Therefore, SAP recommends using the same system default of the column-store configuration for the source and the target system.

When using the column-store, a BW e-fact table has two indexes on the database:

- P-index:
  Unique, clustered b-tree index on all (up to 16) dimensions of the cube

- CS-index:
  Non-clustered column-store index on all fields (dimensions and key figures)

Index Layout in SAP DDIC

When using the column-store, the additional CS-index is created in SAP DDIC. The obsolete single column indexes are not deleted in SAP DDIC. They are simply marked as *Not for SQL Server* (**E**xclude **MSS**).

Index ID	Short text	Unique	DB index name	DB status	Inclusion/ Exclusion	Database system
020	Index using Dimension COND_C01T			D	E	MSS
040	Index using Dimension COND_C011			D	E	MSS
050	Index using Dimension COND_C012			D	E	MSS
060	Index using Dimension COND_C013			D	E	MSS
070	Index using Dimension COND_C014			D	E	MSS
080	Index using Dimension COND_C015			D	E	MSS
090	Index using Dimension COND_C016			D	E	MSS
100	Index using Dimension COND_C017			D	E	MSS
110	Index using Dimension COND_C018			D	E	MSS
CS	**Columnstore Index on ALL fields**		**/BIC/ECOND_C01~CS**			
P	Unique Index (Key Fields) for Table /BIC/ECOND_C01	X	/BIC/ECOND_C01~P			

Index layout on database with CS-index

By choosing *Database Object* from menu *Utilities* in SAP transaction SE11, you can see the indexes on the database. The following example shows the indexes of an e-fact table using the column-store. Here you see up to 16 columns of the table as part of the CS-index. However, in reality, the CS-index contains all 21 columns of the table. SAP does not show the additional index columns here to make sure that the consistency check with SAP DDIC works fine.

Fields: Consistent with the runtime object

Fld Name	Position	Data Type	Length	Decimals	Not null	Default
KEY_COND_C01P	1	int	10		X	0
KEY_COND_C01T	2	int	10		X	0
KEY_COND_C01U	3	int	10		X	0
KEY_COND_C011	4	int	10		X	0
KEY_COND_C012	5	int	10		X	0
KEY_COND_C013	6	int	10		X	0
KEY_COND_C014	7	int	10		X	0
KEY_COND_C015	8	int	10		X	0
KEY_COND_C016	9	int	10		X	0
KEY_COND_C017	10	int	10		X	0
KEY_COND_C018	11	int	10		X	0
D_COSTVALS	12	decimal	17	2	X	0
D_DOCITEMS	13	decimal	17	3	X	0
D_DOCUMENT	14	decimal	17	3	X	0
D_GR_WT_KG	15	decimal	17	3	X	0
D_NETVAL_S	16	decimal	17	2	X	0
D_NT_WT_KG	17	decimal	17	3	X	0
D_OORQTYBM	18	decimal	17	3	X	0
D_OORVALSC	19	decimal	17	2	X	0
D_QUANT_B	20	decimal	17	3	X	0
D_VOLUMCDM	21	decimal	17	3	X	0

Indexes: Consistent with DDIC

Unique index /BIC/ECOND_C01~P
KEY_COND_C01T
KEY_COND_C011
KEY_COND_C012
KEY_COND_C013
KEY_COND_C014
KEY_COND_C015
KEY_COND_C016
KEY_COND_C017
KEY_COND_C018
KEY_COND_C01U
KEY_COND_C01P

Index /BIC/ECOND_C01~CS
KEY_COND_C01P
KEY_COND_C01T
KEY_COND_C01U
KEY_COND_C011
KEY_COND_C012
KEY_COND_C013
KEY_COND_C014
KEY_COND_C015
KEY_COND_C016
KEY_COND_C017
KEY_COND_C018
D_COSTVALS
D_DOCITEMS
D_DOCUMENT
D_GR_WT_KG
D_NETVAL_S

Index layout on database without CS-index

A similar cube without column-store index looks like this:

Indexes: Consistent with DDIC

Index /BIC/ECOPY_BPE~020
KEY_COPY_BPET

Index /BIC/ECOPY_BPE~040
KEY_COPY_BPE1

Index /BIC/ECOPY_BPE~050
KEY_COPY_BPE2

Index /BIC/ECOPY_BPE~060
KEY_COPY_BPE3

Index /BIC/ECOPY_BPE~070
KEY_COPY_BPE4

Index /BIC/ECOPY_BPE~080
KEY_COPY_BPE5

Index /BIC/ECOPY_BPE~090
KEY_COPY_BPE6

Index /BIC/ECOPY_BPE~100
KEY_COPY_BPE7

Index /BIC/ECOPY_BPE~110
KEY_COPY_BPE8

Unique index /BIC/ECOPY_BPE~P
KEY_COPY_BPET
KEY_COPY_BPE1
KEY_COPY_BPE2
KEY_COPY_BPE3
KEY_COPY_BPE4
KEY_COPY_BPE5
KEY_COPY_BPE6
KEY_COPY_BPE7
KEY_COPY_BPE8
KEY_COPY_BPEU
SID_0CALMONTH

Customer Experience with SAP BW using the column-store

Using the column-store has many benefits. The disk space requirements are much lower and BW query performance is consistently better. Furthermore, BW administration is simplified, since you typically do not need to maintain BW aggregates anymore.

Space Savings

SQL Server 2008 introduced row and page compression, which already reduced the disk space usage for SAP BW tables. By using the column-store, you can further decrease the disk space usage of SAP BW e-fact tables by additional 50% to 80% (even compared to page compression). The actual space savings depend on two factors:

- The number of dimensions in the cube
  When using the column-store, all single column indexes (one per dimension) can be dropped. The more dimensions you have, the better disk space savings you see.
- Cardinality of the dimensions
  The column-store compresses data much better compared to the row-store. The compression ratio depends on the cardinality of the dimension (the number of rows in the dimension table).

We saw consistent space savings on our own test systems and on customer systems:

Cube	Data Type	Area	Rows	#Dim	Page comp.	Column-store	Saving
PERF_CSE	generated	Sales & Distribution	100,000,000	11	15.10 GB	6.61 GB	56%
CCA_C001	customer	Cost Center	21,000,000	11	2.70 GB	0.57 GB	79%
PY_C001	customer	Payroll	17,000,000	8	1.74 GB	0.39 GB	77%
PS_C001	customer	Projects	10,000,000	11	1.54 GB	0.38 GB	76%
GL_C10A	customer	General Ledger	16,000,000	7	1.42 GB	0.52 GB	63%
0TCT_C02	customer	OLAP Statistics	49,000,000	15	9.43 GB	1.89 GB	80%

Query Performance

SQL query performance is consistently increased when using the column-store with SAP BW. However, the degree of performance increase of the column-store varied. We saw a wide spread, dependent on the data and type of query. Some queries were up to 50 times faster, while others hardly benefitted from the column-store index. The majority of the SQL queries we tested, ran about 3 to 6 times faster. You can benefit most from column-store indexes under the following conditions:

- SQL Server has enough memory to keep the column-store index (of the selected table) in RAM
- SQL Server has at least a few idle CPU threads to use parallelism
- The selected table has at least a few million rows

Typically, an SAP BW query consists of several SQL queries running in parallel. One SQL query runs against the f-fact table while another SQL query runs in parallel against the e-fact table. When using BW multi-providers, you see additional parallel running queries. Therefore, SAP BW is configuring SQL Server intra-query parallelism (max degree of parallelism) very conservatively: Only two CPU threads are used per SQL query (however, you can change the default configuration using RSADMIN parameter MSS_MAXDOP_QUERY).

For example, a BW query running on a multi provider of four basis cubes, runs eight SQL queries (on four e-fact and four f-fact tables) in parallel. Each of them is using two CPU threads. Therefore, you leverage at least 16 CPU threads when running this single BW query.

In addition to the SQL Server data cache, there is another important cache, the OLAP cache. SAP BW typically caches the results of BW queries on its own. Therefore, many BW queries do not need to access the database, or at least do not have to retrieve all data from the database. For this kind of queries you can hardly see any benefit from the column-store, even if the SQL queries are faster by multiple factors. However, if the BW OLAP cache is not filled, you clearly see a performance improvement.

Aggregates

By creating a fully optimized aggregate, you can enormously speed up the runtime for a particular BW query. However, customers often try to avoid the overhead of creating BW aggregates. If aggregates are really needed, customers try to find a compromise between the overhead of aggregate maintenance and BW query performance: A single BW aggregate is typically not fully optimized for a single BW query. In return, the same aggregate can support a variety of BW queries at the same time.

When using the column-store, you typically do not need aggregates at all. Therefore, you should deactivate the aggregates of a cube at the same time when defining the column-store (in report MSSCSTORE, see below). The definition of the aggregates then still exists in SAP BW. However, the database tables of the aggregates are dropped.

Note: We recommend not using BW aggregates for cubes using the column-store for the following reasons:

1. BW aggregates never use the column-store. This is by design in to support the BW Change Run
2. The BW OLAP engine is not aware of the column-store. Therefore, the OLAP engine may decide to use an aggregate although the basis cube has a column-store index
3. Typically, queries using BW aggregates are slower than queries using a column-store index
4. Not using aggregates results in further disk space savings and less administration efforts

BW compression

Since the non-clustered column-store index is read-only, it has to be dropped at the beginning of the BW cube compression and re-created at the end of the cube compression. This is all performed automatically. A BW administrator has not to take care of this, with one exception: If the BW cube compression fails for any reason (for example due to a full database transaction log), the column-store index may not exist anymore on the database. In this case, the index has to be created manually using SAP transaction RSA1 (see below at “Repair indexes”).

You may argue that the BW cube compression takes more time when using the column-store, since you have to re-create the column-store index. However, it is often the other way round. You do not need to maintain all the single column indexes of the e-fact table when compressing a cube with column-store index. The index creation of a column-store index is very fast. By default, SAP BW is requests eight SQL Server threads for this task. You can further increase the parallelism for index creation by setting the RSADMIN parameter MSS_MAXDOP_INDEXING (see below).

The BW cube compression on Microsoft SQL Server is today much faster than a few years ago. By using the SQL MERGE statement, the aggregation from the f-fact to the e-fact table has become faster. The deletion of the rows from the f-fact table has become much faster due to an improved partition drop algorithm in SQL Server 2008 SP2 and newer SQL Server releases.

However, the cube compression nowadays contains an additional step. It runs an update statistics on all database tables of the compressed BW cube. You can turn off the update statistics by setting a RSADMIN parameter. However, we do not recommend doing so. For details, see http://blogs.msdn.com/b/saponsqlserver/archive/2012/08/20/how-update-statistics-really-works-in-sap-bw.aspx

Note: BW query performance is best, if all BW request are compressed and therefore the column-store index on the e-fact table can be fully leveraged. However, you do not need to compress a cube each time a new request is loaded into the cube. You can schedule BW cube compression once a day or even once a week. Thereby you make sure that the majority of the BW request is always compressed. This is typically sufficient to get a good BW query performance.

Configuring Column-Store in SAP BW

To use the column-store for SAP BW cubes, you first have to define the cubes, which should have a column-store index on the e-fact table. This definition is stored in an SAP Data Dictionary table. The next step is to convert the e-fact table by dropping unnecessary B-Tree indexes and creating the column-store index. You can perform both steps either separately or within a single job using SAP report MSS*CSTORE*.

Defining cubes with Column-Store index

The Data Dictionary table MSS*STORAGE* contains SQL Server specific storage parameters and global settings. For each e-fact table using the column-store, there is a row in table MSSSTORAGE. You can use report MSSCSTORE to change the settings in table MSSSTORAGE.

Note: Table MSSSTORAGE contains additional rows that are not related to the column-store. Do **not** modify the content of table MSSSTORAGE manually. Always use report MSSCSTORE.

Converting cube according to definition

Defining the column-store index for an SAP BW cube is very fast, since it only modifies the SAP Data Dictionary. Actually converting a cube can take some time. Therefore, we recommend running the conversion always as a batch job. During the conversion to the column-store, several b-tree indexes are dropped and a column-store index is created. When converting back to the row-store, the b-tree indexes have to be re-created. Actually, converting the cube is nothing else than repairing the indexes of e-fact table according to the definition stored in table MSSSTORAGE.

Note: Creating indexes, in particular b-trees, requires a significant amount of system resources: CPU, memory, I/O and data and log space on the database. In a typical BW system, process chains drop and create indexes the whole day. To reduce the required transaction log space, it might be a good idea to set the SQL Server recovery model to Bulk-logged. Keep this in mind when setting up your backup strategy for the database of your BW system.

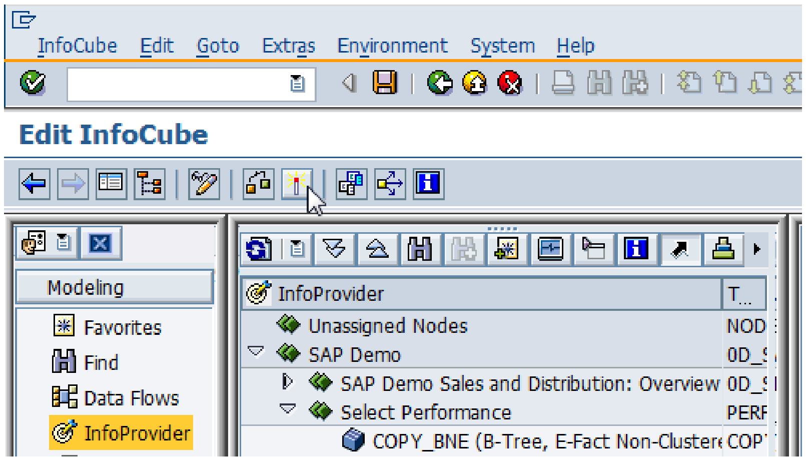
There are several ways to convert a BW cube from row-store to column-store, and vice versa:

Report MSSCSTORE

Using SAP report MSSCSTORE is the easiest way to convert the BW cube. SAP strongly recommends running the report as a batch job. MSSCSTORE is described in detail below.

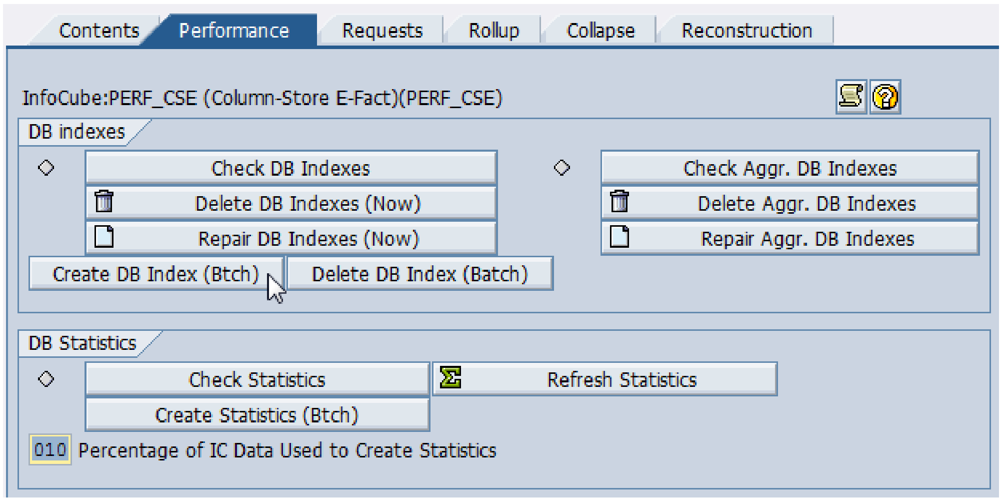
Activate empty cube

If an SAP BW cube is empty, a cube activation creates all database tables and indexes dependent on the configuration in MSSSTORAGE. Therefore, you can simply convert an empty cube by activating it in SAP transaction RSA1.



Repair indexes

You can repair the indexes of all tables of a cube in SAP transaction RSA1 -> *Manage* -> *Performance*. As a matter of course, the index repair takes into account the definition in table MSSSTORAGE.



There are two buttons that both start a batch job for the index repair:

- Repair DB Indexes (Now)
- Create DB Index (Btch)

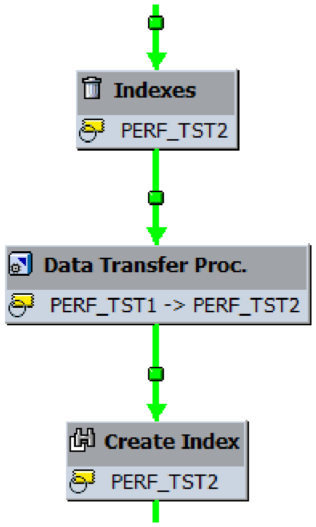
There is only one difference between the two buttons. Pressing the first button starts the batch job immediately. When pressing the second button, you can schedule the batch job at any time.

BW Process chain

Almost all processes in SAP BW are running as batch jobs within BW process chains. Most customers include the process types "Delete Index" and "Generate Index" in their process chains. The process type "Generate Index" performs an index repair. It therefore automatically converts the cube from row-store to column-store (and vice versa), dependent on the current configuration in MSSSTORAGE.

Therefore, the easiest way to convert a cube to column-store is the following:

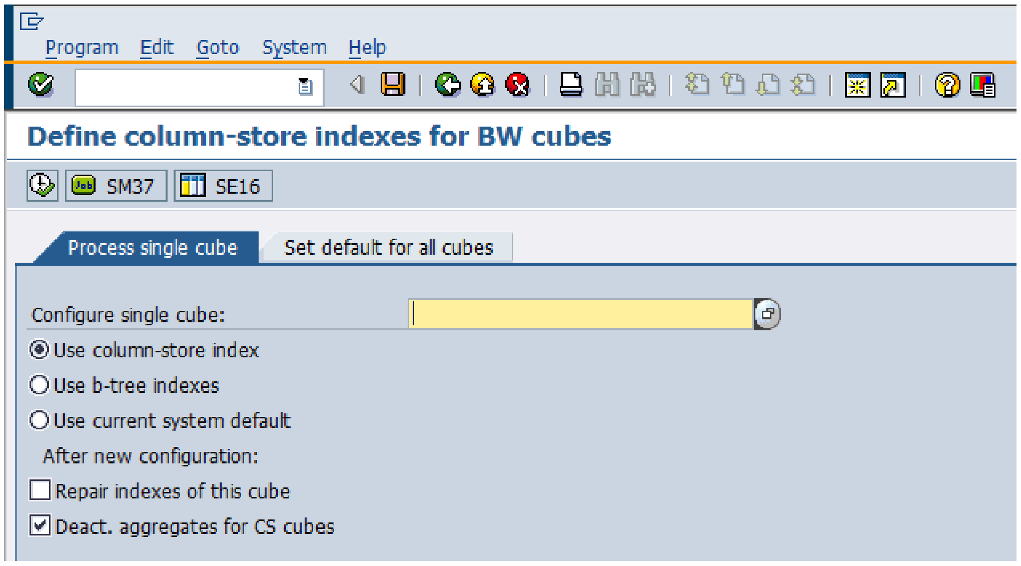
1. Configure the cube for using the column-store in report MSSCSTORE
2. Convert the cube automatically during the next process chain (at night) that contains the process type "Generate Index" for this cube.



Report MSSCSTORE

You can start SAP report MSSCSTORE using SAP transaction SE38. With this report you can:

- Check the content of table MSSSTORAGE
- Configure single cubes or set the system default for using the column-store
- Deactivate BW aggregates of a single cube or all cubes
- Repair the indexes of a single cube or all cubes
- Schedule a batch job for all activities

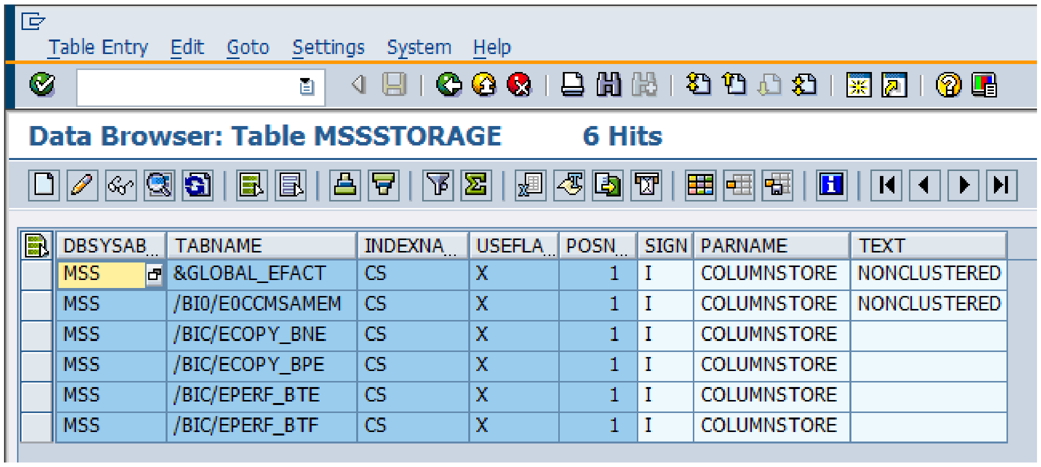


Define column-store indexes for BW cubes

Configure single cube:
(●) Use column-store index
( ) Use b-tree indexes
( ) Use current system default
After new configuration:
[ ] Repair indexes of this cube
[✓] Deact. aggregates for CS cubes

Current configuration

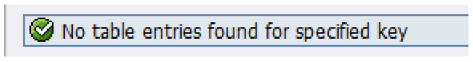
By pressing the button SE16, you can start the *Data Browser* (SAP transaction SE16) for table MSSSTORAGE:



Data Browser: Table MSSSTORAGE 6 Hits

DBSYSAB...	TABNAME	INDEXNA...	USEFLA...	POSN...	SIGN	PARNAME	TEXT
MSS	&GLOBAL_EFACT	CS	X	1	I	COLUMNSTORE	NONCLUSTERED
MSS	/BI0/E0CCMSAMEM	CS	X	1	I	COLUMNSTORE	NONCLUSTERED
MSS	/BIC/ECOPY_BNE	CS	X	1	I	COLUMNSTORE	
MSS	/BIC/ECOPY_BPE	CS	X	1	I	COLUMNSTORE	
MSS	/BIC/EPERF_BTE	CS	X	1	I	COLUMNSTORE	
MSS	/BIC/EPERF_BTF	CS	X	1	I	COLUMNSTORE	

The output is already filtered. It only contains rows that belong to the column-store configuration (PARNAME = "COLUMNSTORE"). If there is no single configuration regarding the column-store in table MSSSTORAGE yet, the Data Browser does not start. Instead, the following status message appears:



No table entries found for specified key

The column-store configuration can either be an *explicit configuration* for a particular cube or an *implicit configuration* using the system default. Valid configuration values are:

- **"NONCLUSTERED"**
  for cubes having a non-clustered column-store index
- **" "** (an empty field)
  for cubes not having a column-store index

Note: You may also see in SE16 the configuration value "NOT_SUPPORTED". This indicates that a cube was explicitly configured to use the column-store. However, it cannot use it, because it includes a key figure with high precision.

The configuartion values are stored in table MSSSTORAGE in field TEXT.The field TABNAME contains the name of the e-fact table for an explicit column-store configuration of a cube. The system default is also stored in table MSSTORAGE with TABNAME = **"&GLOBAL_EFACT"**

In the example above, all cubes are configured implicitly to use the column-store, except for the following four cubes: COPY_BNE, COPY_BPE, PERF_BTE, PERF_BTF.

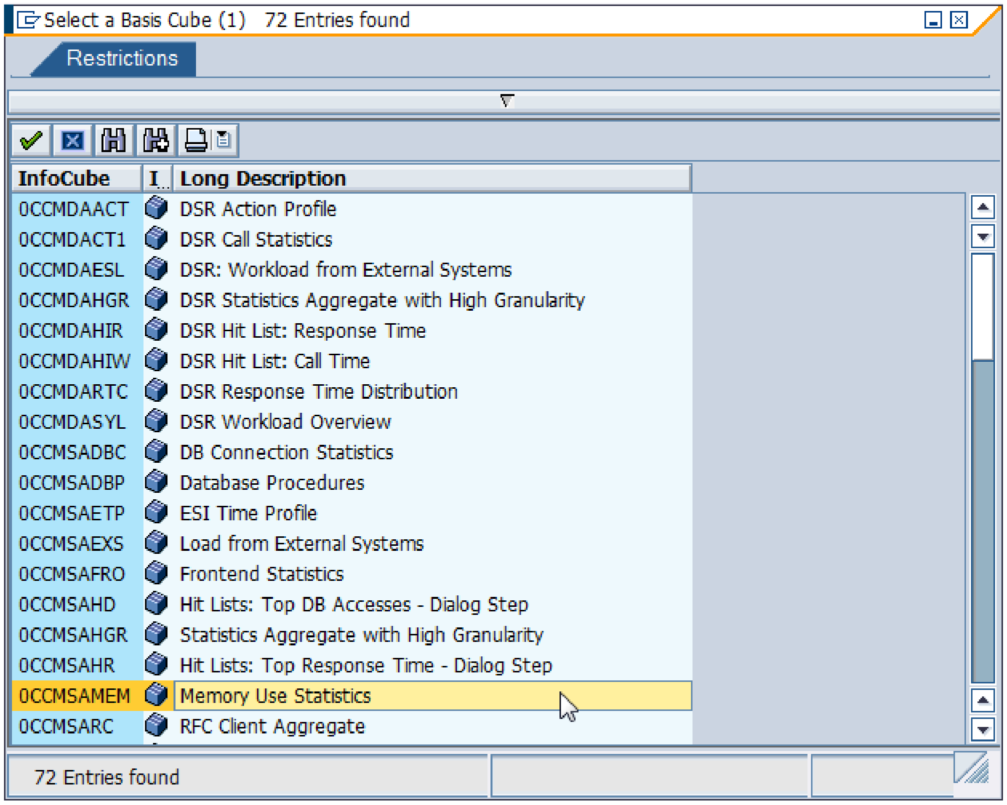
Cube *0CCMSAMEM* (with e-fact table /BI0/E*0CCMSAMEM*) is configured to use the column-store index due to its explicit configuration in addition to the system default. It is useful to store the explicit configuration to keep the cube unchanged, even if you will change the system default in the future.

If there is no row with TABNAME = "&GLOBAL_EFACT" in MSSSTORAGE, then the system default is, not to use the column-store index.

By pressing F3 (green arrow back), you can return to the main screen of report MSSCSTORE

Configure single cube

In the tab *Process single cube* you have to enter the cube name (not the name of the e-fact table) that you want to configure. To get a list of all cubes, you can use the F4 help. Alternatively, you can type the cube name manually in the input field.



InfoCube	I...	Long Description
0CCMDAACT		DSR Action Profile
0CCMDACT1		DSR Call Statistics
0CCMDAESL		DSR: Workload from External Systems
0CCMDAHGR		DSR Statistics Aggregate with High Granularity
0CCMDAHIR		DSR Hit List: Response Time
0CCMDAHIW		DSR Hit List: Call Time
0CCMDARTC		DSR Response Time Distribution
0CCMDASYL		DSR Workload Overview
0CCMSADBC		DB Connection Statistics
0CCMSADBP		Database Procedures
0CCMSAETP		ESI Time Profile
0CCMSAEXS		Load from External Systems
0CCMSAFRO		Frontend Statistics
0CCMSAHD		Hit Lists: Top DB Accesses - Dialog Step
0CCMSAHGR		Statistics Aggregate with High Granularity
0CCMSAHR		Hit Lists: Top Response Time - Dialog Step
0CCMSAMEM		Memory Use Statistics
0CCMSARC		RFC Client Aggregate

72 Entries found

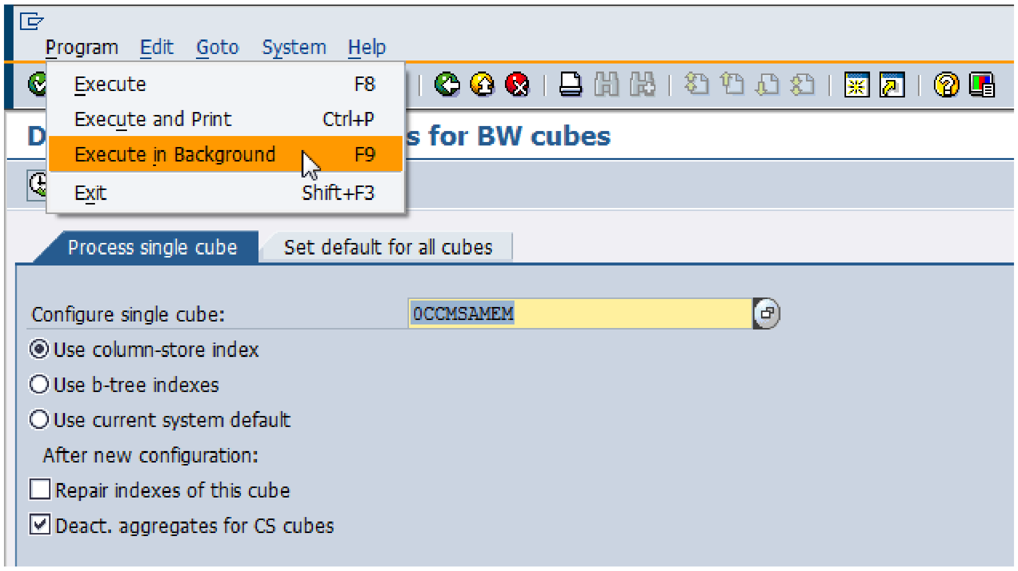
The next step is to choose the column-store configuration for the given cube by pressing one of these radio buttons:

- **Use Column-Store index**
  Creates the explicit configuration: *use the column-store* for the cube.
  Technically, it adds an entry in MSSTORAGE for the e-fact table of this particular cube with PARNAME = "COLUMNSTORE" and TEXT = "NONCLUSTERED"
- **Use B-Tree indexes**
  Creates the explicit configuration*: use not the column-store* for the cube.
  Technically, it adds an entry in MSSTORAGE for the e-fact table of this particular cube with PARNAME = "COLUMNSTORE" and TEXT = " "
- **Use current system default**
  Deletes the explicit configuration for the cube.
  Technically, it deletes the entry in MSSSTORAGE for the e-fact table of this particular cube

In addition, you can select the following check boxes:

- **Deact. aggregates for CS cubes**
  If the configuration results in using the column-store (by either an explicit configuration or the system default), then the BW aggregates of this particular cube are deactivated. They are still defined in SAP BW. However, the database tables of the aggregates are dropped on SQL Server, which results in further disk space savings.
- **Repair indexes of this cube**
  Repairs all indexes of the chosen cube according to the configuration in MSSSTORAGE

To start the process, you have to press the button *Execute* (F8). However, when selecting one of the two options above, it is strongly recommended starting the process as a batch job. Therefore, you have to choose *Execute in Background* (F9) from the *Program* menu.



Configure single cube: 0CCMSAMEM

(●) Use column-store index
( ) Use b-tree indexes
( ) Use current system default

After new configuration:

☐ Repair indexes of this cube
☑ Deact. aggregates for CS cubes

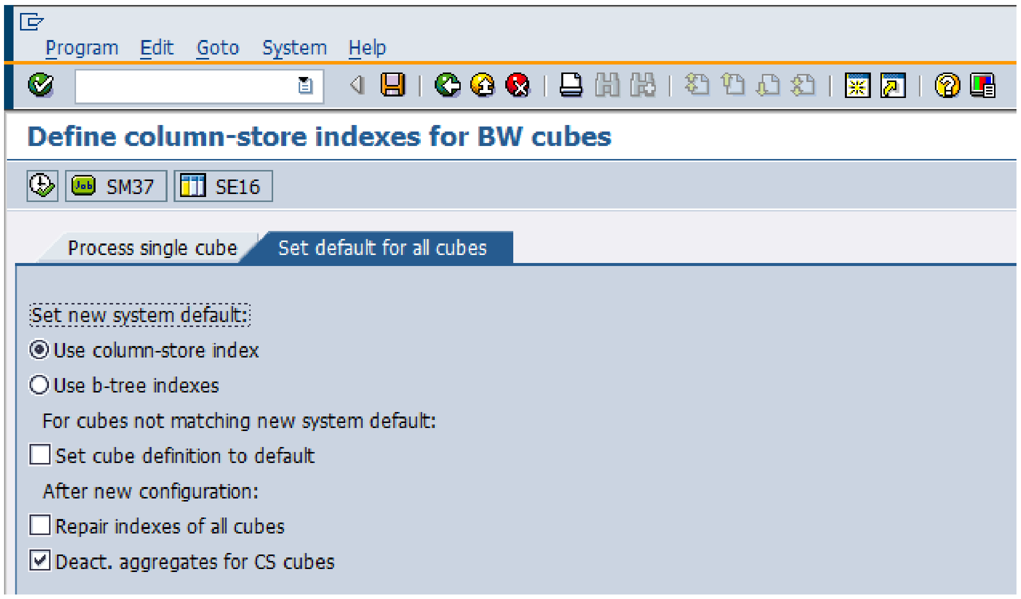
To check the status of the background job and to view the job log files, use SAP transaction SM37. Alternatively, you can press the button *SM37*. In this case you can return to the main screen of report MSSCSTORE by pressing *F3* (green arrow back).

Note: The main screen of report MSSCSTORE has two tabs: *Process single cube* and *Set default for all cubes.* When executing the report (in dialog or background), the outcome depends on the currently visible tab. To configure a single cube, you must execute the report while tab *Process single cube* is visible. To change the system default, the tab *Set default for all cubes* has to be the active tab.
When creating a report variant, you can set parameters for both tabs at the same time. Therefore, SAP does not recommend creating variants manually for report MSSCSTORE. Instead, you can simply press F9 to schedule the report with an automatically created, temporary variant that uses the chosen options on the active tab.

Configure system default

In the tab *Set default for all cubes* you can configure the default for all new BW cubes. You can use one of the following options:

- **Use Column-Store index**
  Use the column-store for cubes that do not have an explicit configuration.
  Technically, it adds/updates the entry in MSSTORAGE for TABNAME = "&GLOBAL_EFACT" with PARNAME = "COLUMNSTORE" and TEXT = "NONCLUSTERED"
- **Use B-Tree indexes**
  Do not use the column-store for cubes that do not have an explicit configuration.
  Technically, it adds/updates the entry in MSSTORAGE for TABNAME = "&GLOBAL_EFACT" with PARNAME = "COLUMNSTORE" and TEXT = ""



Define column-store indexes for BW cubes

Set new system default:
(●) Use column-store index
( ) Use b-tree indexes
For cubes not matching new system default:
[ ] Set cube definition to default
After new configuration:
[ ] Repair indexes of all cubes
[✓] Deact. aggregates for CS cubes

In addition to the system default, you may also modify existing explicit configurations at the same time:

- **Set cube definition to default**
  Overwrites existing explicit configurations with the new default value. As a result, all cubes are configured identically.

Note: SAP BW automatically creates an explicit column-store configuration entry (in table MSSSTORAGE) based on the system default, each time the indexes of a cube are checked or created. This implementation ensures that already existing cubes keep their column-store configuration once you change the system default. As a result, table MSSSTORAGE might contain hundreds of rows.

The options **Deact. aggregates for CS cubes** and **Repair indexes of all cubes** work the same as described for the single cube configuration. However, the options have effect on all cubes of the system, even if their column-store configuration was not changed.

Note: There is no option in report MSSCSTORE to activate aggregates when configuring the cubes to use b-tree indexes. This decision was made for two reasons: Firstly, customers often have defined some aggregates, which are deactivated by default. Secondly, creating the aggregates can take a significant amount of time. Therefore, customers should schedule the aggregate creation using a process chain, if needed.

Administration

Configuring Resources

To ensure optimal system performance, you have to give sufficient system resources to SQL Server. The most important resources are memory and CPU. For using the column-store with SAP BW, you should have a database server with at least eight CPU threads (for example, 2 CPUs having 4 cores per CPU). In typical customer scenarios, the database server has 16 CPU threads or more.

SQL Server memory

SQL Server memory manager dynamically allocates memory used for the column-store. An administrator only has to make sure that the total memory configured for SQL Server is sufficient. Normal tables and b-tree indexes allocate memory from SQL Server *Buffer Pool*. The column-store uses its own memory pool, the *Column Store Object Pool*. You can check the size of both pools by running the following SQL command:

- `select * from sys.dm_os_memory_broker_clerks`

The query returns 2 rows containing the total size of both pools (or only one row for the Buffer Pool, if the Column Store Object Pool is not used at all).

As of SQL Server 2012, the configuration options *min* and *max server memory* contain all memory used by SQL Server, not only the Buffer Pool. For more information about SQL Server 2012 memory configuration for SAP systems, see SAP note 1702408.

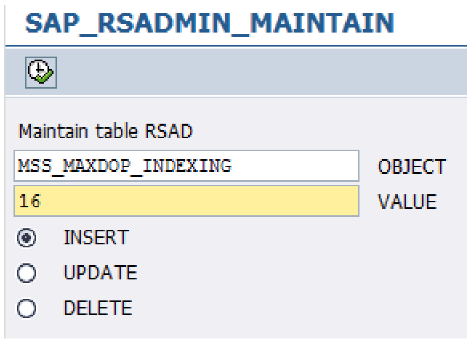
SQL Server parallelism

SAP recommends setting the global SQL Server configuration option max degree of parallelism to 1. SAP BW overwrites this global setting by means of SQL Server optimizer hints. Per default, SAP BW requests two SQL Server threads per query and eight SQL Server threads for index creation. However, you can change the default behavior by setting RSADMIN parameters:

- MSS_MAXDOP_INDEXING
  This RSADMIN parameter sets the maximum number of CPU threads that are used during index creation in SAP BW (for column-store indexes and b-trees). Index creation is performed when originally creating the column-store index, during BW process chains, and during cube compression. The default value of MSS_MAXDOP_INDEXING is 8. SAP does not recommend decreasing this value. However, you may want to increase it, if your database server has more than 16 CPU threads.

- MSS_MAXDOP_QUERY
  This RSADMIN parameter sets the maximum number of CPU threads that are used for a single SQL query as part of a BW query. The default value of MSS_MAXDOP_INDEXING is 2. The degree of parallelism was chosen relatively low, since a single BW query typically results in many parallel running SQL queries. You may want to increase MSS_MAXDOP_QUERY (to 3, 4 or 6), if your database server has more than 16 CPU threads. However, you should never decrease it below 2.

The easiest way to set a RSADMIN parameter is to use SAP report SAP_RSADMIN_MAINTAIN. SAP strongly recommends using this report, because table RSADMIN is buffered on the SAP application servers.


SAP_RSADMIN_MAINTAIN

Maintain table RSAD

MSS_MAXDOP_INDEXING	OBJECT
16	VALUE

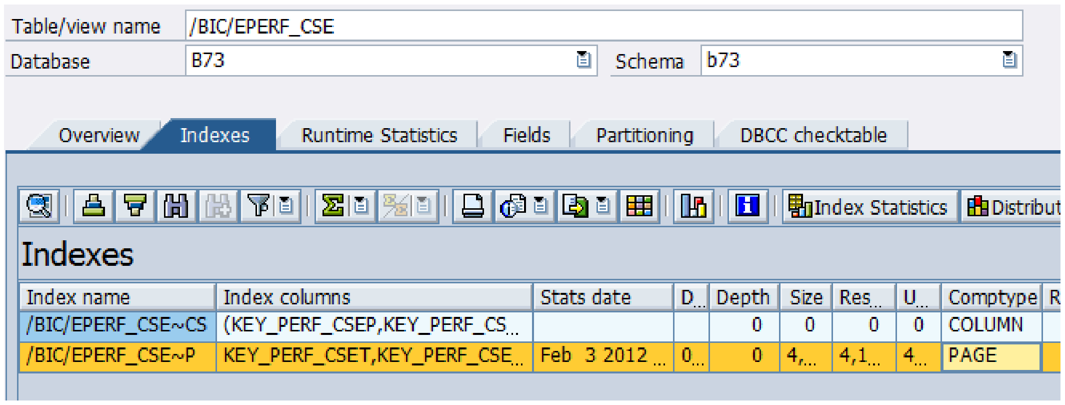
[✓ INSERT] UPDATE DELETE

Checking column-store property

Using report MSSCSTORE, you can configure the column-store property of a BW cube in SAP DDIC. At the same time, you can create the column-store index on the database (repair the indexes of the cube). However, you cannot check the actual status of the indexes here. For this purpose you can use different SAP reports:

DBACOCKPIT

When checking the database indexes of the e-fact table in SAP transaction DBACOCKPIT (as of SAP NetWeaver 7.3), you can see the column-store as a new compression type: COLUMN



Table/view name	/BIC/EPERF_CSE		
Database	B73	Schema	b73

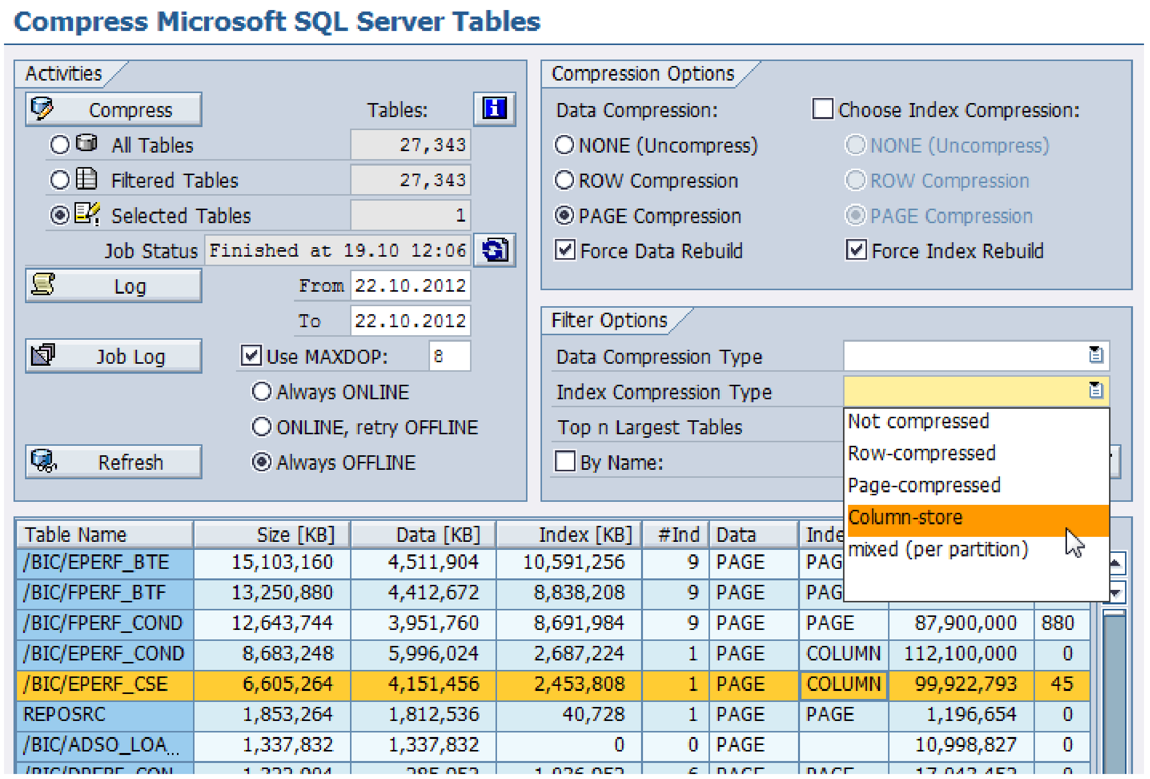
Overview | Indexes | Runtime Statistics | Fields | Partitioning | DBCC checktable

Indexes

Index name	Index columns	Stats date	D..	Depth	Size	Res..	U..	Comptype	R
/BIC/EPERF_CSE~CS	(KEY_PERF_CSEP,KEY_PERF_CS...			0	0	0	0	COLUMN	
/BIC/EPERF_CSE~P	KEY_PERF_CSET,KEY_PERF_CSE...	Feb 3 2012 ...	0...	0	4,...	4,1...	4...	PAGE	

MSSCOMPRESS

Report MSSCOMPRESS is used to change the compression type of database indexes. This report has been extended to get an overview of all your column-store indexes in the system. You can filter for tables that have a column-store index, and check the index size. However, you cannot use this report to change the column-store property of a table, since COLUMN is technically not simply a compression type. For this you have to run report MSSCSTORE, which deletes the needless b-tree indexes on the database and in SAP DDIC when creating the column-store index.



Compress Microsoft SQL Server Tables

Activities
- Compress — Tables:
- All Tables: 27,343
- Filtered Tables: 27,343
- [✓ Selected Tables]: 1
- Job Status: Finished at 19.10 12:06
- Log — From 22.10.2012 To 22.10.2012
- Job Log — [✓] Use MAXDOP: 8
- Always ONLINE / ONLINE, retry OFFLINE / [✓ Always OFFLINE]
- Refresh

Compression Options
- Data Compression: NONE (Uncompress) / ROW Compression / [✓ PAGE Compression]
- [✓] Force Data Rebuild
- Choose Index Compression: NONE (Uncompress) / ROW Compression / [✓ PAGE Compression]
- [✓] Force Index Rebuild

Filter Options
- Data Compression Type
- Index Compression Type: Not compressed / Row-compressed / Page-compressed / [✓ Column-store] / mixed (per partition)
- Top n Largest Tables
- By Name:

Table Name	Size [KB]	Data [KB]	Index [KB]	#Ind	Data	Inde		
/BIC/EPERF_BTE	15,103,160	4,511,904	10,591,256	9	PAGE	PAG	[illegible]	[illegible]
/BIC/FPERF_BTF	13,250,880	4,412,672	8,838,208	9	PAGE	PAG	[illegible]	[illegible]
/BIC/FPERF_COND	12,643,744	3,951,760	8,691,984	9	PAGE	PAGE	87,900,000	880
/BIC/EPERF_COND	8,683,248	5,996,024	2,687,224	1	PAGE	COLUMN	112,100,000	0
/BIC/EPERF_CSE	6,605,264	4,151,456	2,453,808	1	PAGE	COLUMN	99,922,793	45
REPOSRC	1,853,264	1,812,536	40,728	1	PAGE	PAGE	1,196,654	0
/BIC/ADSO_LOA...	1,337,832	1,337,832	0	0	PAGE		10,998,827	0

Related Content

Column-store indexes, described in SQL Server Books Online:
http://msdn.microsoft.com/en-us/library/gg492088.aspx

xVelocity technology, described in SQL Server Books Online:

http://msdn.microsoft.com/en-us/library/hh922900.aspx

SAP BW related issues in blog "SAP On SQL Server":
http://blogs.msdn.com/b/saponsqlserver/archive/tags/bw/

SAP Community Network "SAP On SQL Server":
http://scn.sap.com/community/sqlserver

Download of SQL Server 2012 CU2 (SQLServer2012_RTM_CU2_2703275_11_0_2325_x64):
http://support.microsoft.com/hotfix/KBHotfix.aspx?kbnum=2703275&kbln=en-us

Copyright