WhereScape®

SQL SERVER DATA COMPRESSION

Technical Paper

www.wherescape.com

WhereScape®

WhereScape®

SQL SERVER DATA COMPRESSION

Document Control

Author

Martin Norgrove, Senior Consultant,
WhereScape Limited.

Change History

Version	Date	Change Description
1.1	15/09/2014	Final Version

SQL Server Data Compression

Overview

Data compression has been around a while in SQL Server, being first introduced in SQL Server 2008, prior to this SQL Server 2005 supported a form of compression using a data type called Vardecimal. Vardecimal is a deprecated feature in SQL Server 2012 enabling this feature in new installations should be carefully considered. Data Compression is an Enterprise only feature.

The data compression feature of SQL Server is implemented on tables, partitions and indexes within a database and can be applied in one of three ways, ROW, PAGE or NONE. NONE being no compression specified; the default setting.

ROW Compression

Stores integer, decimal and string format data in variable length fields (essentially equivalent to the SQL 2005 Vardecimal feature) rather than fixed length. It also reduces the meta-data overhead needed to store the data.

ROW compression will typically yield a smaller space saving than PAGE compression.

PAGE Compression

PAGE compression utilises three separate features to compress the data, firstly ROW compression then prefix compression and finally dictionary compression. Prefix compression looks for repeated values in columns, either partial or complete, and compresses them. After prefix compression has been applied then dictionary compression is applied to further compress any remaining repeated values.

PAGE compression will typically yield a greater level of compression than ROW compression

Compression can be specified at several granularities within a database, you can for example:

- Have a mixture of ROW, PAGE and uncompressed tables with a database.
- Compress a heap or clustered index and have no compression on the non-clustered indexes.
- Have a mixture of ROW, PAGE and uncompressed partitions within a single table.

Benefits

There are two main benefits of utilising compression for database tables:

- **Reduced Physical IO** – There is less data on disk to read when accessing physical data

WhereScape®

- **Reduced Logical IO** – Because the data pages are still compressed in memory, more pages can fit into the available memory making the process more efficient.

Typically savings from both physical and logical IO are greatest when tables and indexes are being scanned.

Evaluating Data Compression

Microsoft have provided a system stored procedure to help in estimating the space that might be saved by the different compression options, essentially it copies a sample of data into TempDb and compresses it before returning the estimated savings.

The procedure is called sp_estimate_data_compression_savings, unfortunately this procedure can only evaluate one table at a time. A sample script used to evaluate several tables at once is included in the appendix.

The amount of space saved by compression per table will be variable and will depend on several factors, including data types and the type of workload the system is being used for. Some data patterns and workloads that typically work well are:

- Integer, numeric or fixed length string data types where the data does not use all the available bytes supported by the data type.
- Columns with a significant number of NULLS
- Columns with a significant volume of repeated values
- Workloads where tables and indexes are frequently scanned

Compression Considerations

The following are a series of considerations when using data compression:

- Data Compression is an Enterprise only feature in SQL Server 2008, 2008R2, 2012 & 2014
- Typical CPU overhead for compression is 10%, as long as there is sufficient CPU capacity on the system then compression should be considered
- The data compression setting of the table will apply to all partitions in the table unless otherwise specified
- Inserting into a compressed table using BULK INSERT will typically be slower than using an uncompressed table
- Compressing a table or index requires additional free space, CPU and IO. Typically there are advantages in considering the following settings:
    - Using a SIMPLE recovery model to minimise transaction log activity
    - Use SORT_IN_TEMPDB = ON when rebuilding indexes, setting utilises Tempdb as the workspace rather than the user database
    - Use the ONLINE = OFF setting when rebuilding indexes, OFFLINE operations tend to use less CPU and are faster (although the table is unavailable during the compression operation
- Typically it is best to compress one table/index at a time
- Consider compressing smaller tables first as the space freed can then be used as workspace for later larger tables without growing the data files
- You can apply compression to a table by rebuilding the clustered index on the table specifying the required compression in the rebuild TSQL

WhereScape®

Recommended Compression Settings

Best practices will vary as hardware and data characteristics will vary from solution to solution, however there are several settings that can be generally recommended:

- If there is more than 10% spare CPU capacity on the system and the SQL edition is Enterprise then Compression should be seriously considered
- Data Warehouse workloads are typically scan intensive and data volumes can be high. In general apply PAGE compression in the following way:
    - Facts in particular contain a high level of repetition so tables, partitions and indexes should be compressed
    - Stage, Data Store and Dimension tables, partitions and indexes have less data and repetition these could be compressed depending on circumstances
    - Load tables where BULK INSERT is often used to load data quickly should remain uncompressed
- Indexes should have "Sort in Tempdb" setting on to minimise data file growth
- Index builds should have ONLINE set to OFF, which is the default setting
- Use the stored procedure sp_estimate_data compression_savings to estimate compression savings
- Implement compression on smaller tables first to save time re-claiming space from data files later
- If possible, check data warehouse processing and query performance with a range of compression settings to ensure the optimum configuration

RED Support for Data Compression

RED supports data compression for SQL Server 2008, 2008R2, 2012 & 2014 Enterprise editions. Data compression features can be found in two areas of RED.

- For Tables and Indexes under Properties Storage
- For Indexes there is an additional feature for "Sort in Tempdb" which can be selected to reduce the space overhead of rebuilding compressed indexes, and offload this to Tempdb

WhereScape®

Appendices

References

The following are several reference sites for further information:

Description	URL
Script for estimating compression savings on multiple tables	http://sqlfool.com/2011/06/estimate-compression-savings/
Data Compression in SQL Server 2012	http://msdn.microsoft.com/en-us/library/cc280449.aspx
Data Compression: Strategy, Capacity Planning and Best Practices whitepaper	http://msdn.microsoft.com/en-us/library/dd894051(v=SQL.100).aspx
Alter Index, syntax for rebuilding indexes and changing compression	http://msdn.microsoft.com/en-us/library/ms188388.aspx
Alter Table, syntax for rebuilding tables and changing compression	http://msdn.microsoft.com/en-us/library/ms190273.aspx
sp_estimate_data_compression_savings	http://msdn.microsoft.com/en-us/library/cc280574.aspx