



*The World's Largest Community
of SQL Server Professionals*



Using SQL Server Compression to Boost Database Performance

Brad M. McGehee

Director of DBA Education

Red Gate Software

www.bradmcgehee.com/presentations

My Assumptions About You

- You may be a part-time Production or Development DBA with at least one year's experience.
- You have a basic understanding of SQL Server internals.
- You want an introduction to SQL Server 2008 Row and Page Compression, and how it can potentially boost SQL Server performance.

What We Are Going to Learn Today

- Introduction to Data Compression
- Row Compression Overview
- Page Compression Overview
- How Data Compression Works
- Compressions Pros and Cons
- Deciding What to Compress
- How to Implement Row Compression
- How to Implement Page Compression
- Data Compression Best Practices

Introduction to Data Compression

- Data compression in SQL Server offers two potential benefits to the DBA:
 - Reducing the size of the physical database files (MDF/NDF), reducing the amount of physical disk storage required, and saving storage costs.
 - Reduce the amount of I/O required for a workload, helping to boost overall performance.

Two Types of Data Compression

- SQL Server 2008 and later (Enterprise Edition) offers two forms of data compression:
 - **Row-level Data Compression:** Row-level data compression is essentially turning fixed length data types into variable length data types. It also stores zero and null values in only 4 bits, saving additional space.
 - **Page-level Data Compression:** Page-level data compression starts with row-level data compression, then adds two additional compression features: prefix and dictionary compression.
- These database objects can be compressed:
 - A table stored as a heap
 - A table stored as a clustered index
 - A non-clustered index
 - An indexed view
 - Partitioned tables and indexes

Row Compression

- Stores **fixed length numeric data types** as if they were variable-length data types.
 - For example, if you store the value 1 in a bigint data type, storage will only take 1 byte, not 8 bytes, which the bigint data types normally takes.
- Storing **CHAR data types** as variable-length data types.
 - For example, if you have a CHAR (100) data type, and only store 10 characters in it, blank characters are not stored, thus reducing the space needed to the store data.
- A few other data types can also be compressed, but not all.
- Uses only 4 bits to store NULL or 0 values.
- Allows more rows to fit on a page.

Page Compression

- It starts out by using row-level data compression to fit as many rows as it can on a single page. Assuming a page is full, then page compression kicks in.
- Next, prefix compression is run. Repeating patterns of data at the beginning of the values of a given column are removed and substituted with an abbreviated reference that is stored in the compression information (CI) structure that immediately follows the page header of a data page.
- And last, dictionary compression is used. Dictionary compression searches for repeated values anywhere on a page and stores them in the CI.

Page Compression—Illustrated

Page Header		
aaabb	aaaab	abcd
aaabcc	bbbb	abcd
aaacc	aaaacc	bbbb

Before
Page Compression

Page Header		
aaabcc	aaaacc	abcd
4b	4b	[empty]
[empty]	[0bbbb]	[empty]
3ccc	[empty]	[0bbbb]

After
Prefix Compression

Page Header		
aaabcc	aaaacc	abcd
4b	[0bbbb]	
0	0	[empty]
[empty]	1	[empty]
3ccc	[empty]	1

After
Dictionary Compression

Illustrations from Books Online

How Data Compression Works

- Compression is handled under the covers by the SQL Server Storage Engine.
- When data is passed to the Storage Engine, it is compressed and stored in a compressed format.
- When the Storage Engine passes the data to another part of SQL Server, such as the Relational Engine, then the Storage Engine automatically uncompresses it.
- This means that other parts of SQL Server don't need to understand compression.
- While this takes extra CPU overhead to accomplish, in many cases, the amount of disk I/O saved by compression more than makes up for the CPU costs, boosting performance.

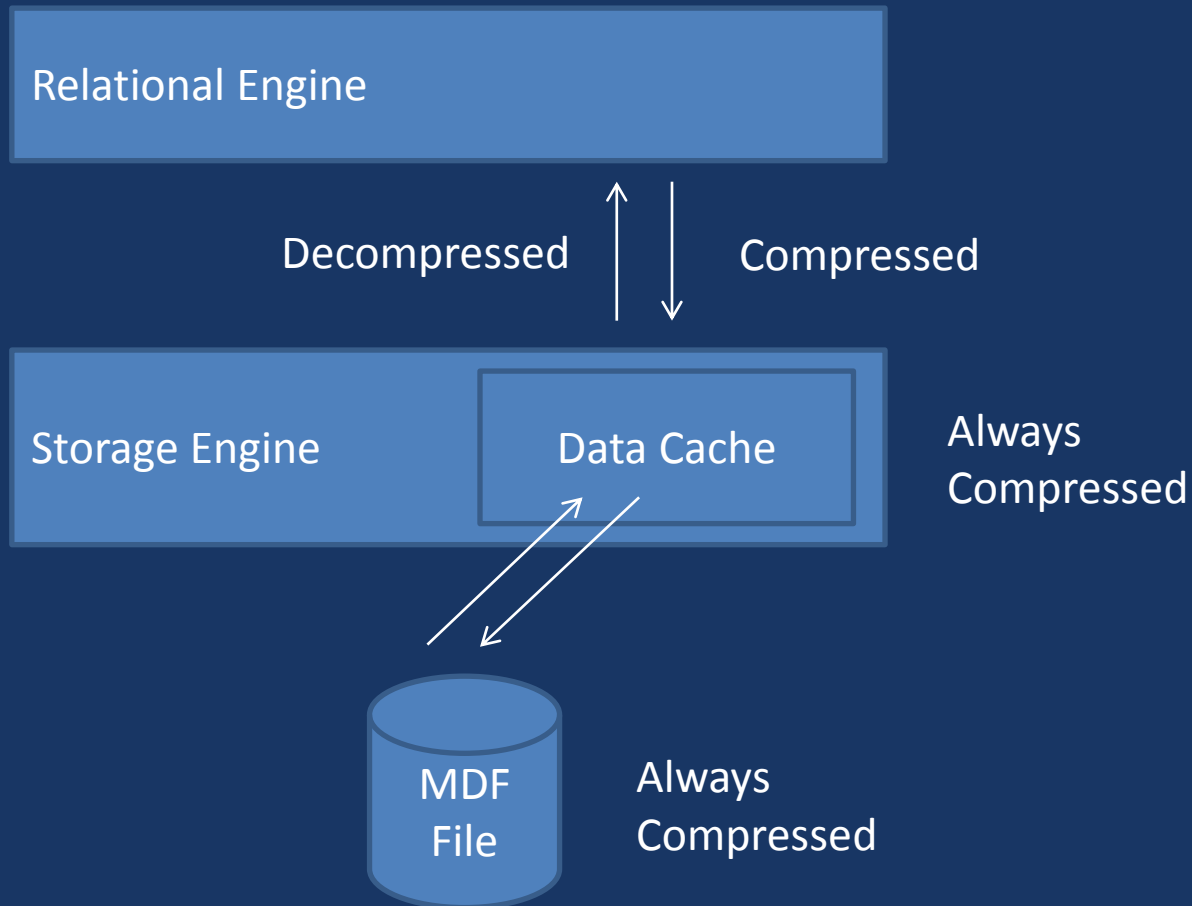
Data Compression Example (1)

- Let's say that we want to update a row in a table, and that the row we want to update is currently stored on disk in a table that is using row-level data compression.
- When we execute the UPDATE statement, the Relational Engine (Query Processor) parses, compiles, and optimizes the UPDATE statement, ready to execute it.
- Before the statement can be executed, the Relational Engine needs the row of data that is currently stored on disk in the compressed format, so the Relational Engine requests the data by asking the Storage Engine to go get it.

Data Compression Example (2)

- The Storage Engine goes and gets the compressed data from disk and brings it into the Data Cache, where the data continues to remain in its compressed format.
- Once the data is in the Data Cache, the row is handed off to the Relational Engine from the Storage Engine. During this pass off, the compressed row (not the entire page) is uncompressed and given to the Relational Engine to UPDATE.
- Once the row has been updated, it is then passed back to the Storage Engine, where it is compressed and stored in the Data Cache. At some point, the page with the modified row will be flushed to disk, where it is stored on disk in its compressed format.

Graphical Overview



Benefits of Using Data Compression

- Compression is in real-time.
- Applications don't need to be rewritten.
- Compression reduces the amount of disk space required to store data (MDF/NDF files are smaller, LDF files are not affected).
- Compression allows more data to be stored in the buffer cache, which means fewer logical and physical I/Os are needed to retrieve data.
- You can choose between Row or Page compression, whichever is best.
- You can choose to compress only those objects you want to compress, thus optimizing compression on an object-by-object basis.
- Data warehousing generally works great with compression, although OLTP databases may also benefit, depending on circumstances.

Disadvantages of Using Data Compression

- Additional CPU use is required to implement compression.
- Data varies greatly in how compressible it is.
- FILESTREAM data can't be compressed.
- Sometimes, the overhead of additional CPU use exceeds the savings in reduced disk I/O, especially for tables with lots of data modifications.
- Tables, and indexes must be compressed separately.
- In most cases, you will want to compress some tables and indexes, but not others. In other words, you don't generally compress everything.
- Rebuilding compressed indexes takes longer than uncompressed indexes.
- Compressed databases may not be restored or reattached to SQL Server instances that are not SQL Server 2008 or higher (Enterprise Edition).
- Other than through testing, it is hard to tell whether compression's benefits exceeds its disadvantages.

Deciding What to Compress

- Factors to help you determine what objects to compress:
 - Estimated space savings
 - Characteristics of the application workload
- By taking each of the above into your calculations, you can determine, on a case-by-case basis, which objects are likely candidates for compression.
- For example, you are looking for objects that are very compressible and have the “right” workload characteristics.

Estimating Space Savings

- Use `sp_estimate_data_compression_savings` to estimate space savings on a table-by-table basis.
- This SP uses TEMPDB to compressed sampled data, one table at a time.
- This SP is resource intensive, so ideally it should only be done on a copy of your production database.

Application Workload

- The “ideal” objects to compress have these two characteristics in common:
 - The lower the number of update operations on an object, the more it will benefit from compression.
 - The higher the percentage of scans on an object, the more it will benefit from compression.
- See [http://msdn.microsoft.com/en-us/library/dd894051\(v=sql.100\).aspx](http://msdn.microsoft.com/en-us/library/dd894051(v=sql.100).aspx) for detailed information on how to calculate this information for your own databases.
- Only by testing can you be sure if compression will work in your circumstances.

How to Implement Compression

- Compression can be turned on when a table or index is first created:
 - CREATE TABLE or CREATE INDEX
- Compression can be turned on anytime after a table or index is created:
 - ALTER TABLE or ALTER INDEX

Turning on Compression Using ALTER TABLE

```
ALTER TABLE table_name  
REBUILD WITH (DATA_COMPRESSION = PAGE)
```

```
ALTER TABLE table_name  
REBUILD WITH (DATA_COMPRESSION = ROW)
```

```
ALTER TABLE table_name  
REBUILD WITH (DATA_COMPRESSION = NONE)
```

Turning on Compression Using ALTER INDEX

```
ALTER INDEX index_name ON table_name  
REBUILD WITH ( DATA_COMPRESSION = ROW)
```

```
ALTER INDEX index_name ON table_name  
REBUILD WITH ( DATA_COMPRESSION = PAGE )
```

```
ALTER INDEX index_name ON table_name  
REBUILD WITH ( DATA_COMPRESSION = NONE)
```

Compression Demo

- Demo Row and Page Compression
 - First Demo: Using SSMS
 - Second Demo: Using T-SQL

Data Compression Best Practices

- Data compression is not always appropriate, especially if tables aren't very compressible, or if subject to heavy data modification.
- Always test compression on a test box, not a production server.
- If your server is already experiencing a CPU bottleneck, then compression may not be a good solution for you.
- Initial data compression is resource intensive, and should be scheduled when the database is not too busy.
- Using compression with table partitions or replication requires special considerations.

Alternative to SQL Server Compression

- If you don't have SQL Server 2008 R2, Enterprise Edition or higher, and you need the benefits of compression (reduced storage space and IO), consider SQL Storage Compress.
- This application will compress the entire database, and is completely invisible to SQL Server, providing you with most of the benefits of native SQL Server compression, without the cost, and without the difficulty of selecting which objects to compress. Works with 2005/2008 and with Standard Edition.
- See this article for more information:

<http://www.simple-talk.com/sql/sql-tools/brads-sure-guide-to-sql-storage-compress/>

Take Aways From This Session

- Data compression has the potential to significantly reduce the amount of disk space and buffer cache needed to store data under the right conditions.
- Data compression has the potential to significantly reduce logical and disk I/O.
- While compression incurs additional CPU overhead, in many cases, the reduction in disk I/O more than compensates for the increased CPU use.
- Determining the benefits of compression is not always clear, and you will need to treat each case separately.
- **Challenge:** When you get back to work, evaluate one of your databases for objects that could be potentially compressed.

Find Out More

Free E-Books:

- www.sqlservercentral.com/Books

Check these out:

- www.SQLServerCentral.com
- www.Simple-Talk.com

Contact me at:

bradmcgehee@hotmail.com

Blogs:

www.bradmcgehee.com

www.twitter.com/bradmcgehee

[Click Here for a free 14-day trial of the Red Gate SQL Server Toolbelt](#)



The World's Largest Community of SQL Server Professionals



*The World's Largest Community
of SQL Server Professionals*

Thanks for Attending

Visit www.sqlservercentral.com for free SQL Server eBooks, articles, videos, blogs, news, and more.

Please Don't Forget to Turn in Your Evaluations