Presentation Summary

The limiting factor in most database systems is the ability to read and write data to the IO subsystem.

We're still using storage layouts and methodologies in SQL Server that are a reflection of old spinning media in times gone by.

Until major changes are made to the internal storage layouts, we have "some" hope with options such as **data compression**, **sparse columns** and **filtered indexes**, which not only save space on disk, but also reflect a saving in memory.

In this session we will go over the IO savings technologies presented in SQL Server, and discuss how implementing some of these will assist in your operational performance goals.

The idea behind compression

- Save on storage costs
  - SAN based storage
  - Backup storage

- Improve Read Performance
  - IO is generally the worst performer in Database infrastructure
  - Lower IO & Higher CPU = reduced transactional turnaround*

* In most cases

Vardecimal Implementation

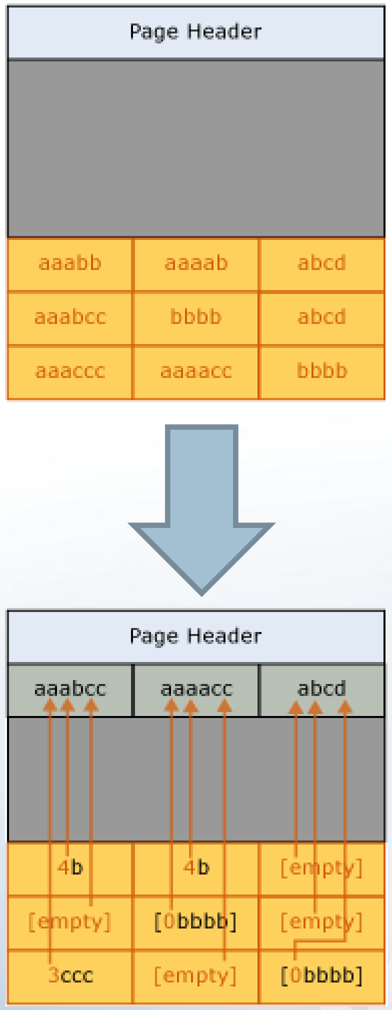
- Introduced in SQL Server 2005 Service Pack 2 (SP2)
- Marked as deprecated shortly thereafter in preference to Row and Page Compression
- Applied to a whole table
- Stored using scientific notation
  - (sign) * mantissa * $10^{exponent}$

Row Compression Implementation

- Only changes the physical storage format of the data that is associated with a data type.
- Uses variable-length storage format for numeric types.
- Stores fixed character strings using variable-length format by not storing the blank characters.
  - NULL and 0 values across all data types are optimized and take no bytes.

Page Compression Implementation

- Does the following:
  1. Row compression
  2. **Prefix compression***
  3. Dictionary compression
  4. Unicode Compression



Prefix Compression:

- Value is identified that can be used to reduce the storage space for the values in each column.
- Row is created to identify the prefix.
- Repeated prefix values in the column are replaced by a reference

Images from SQL BOL

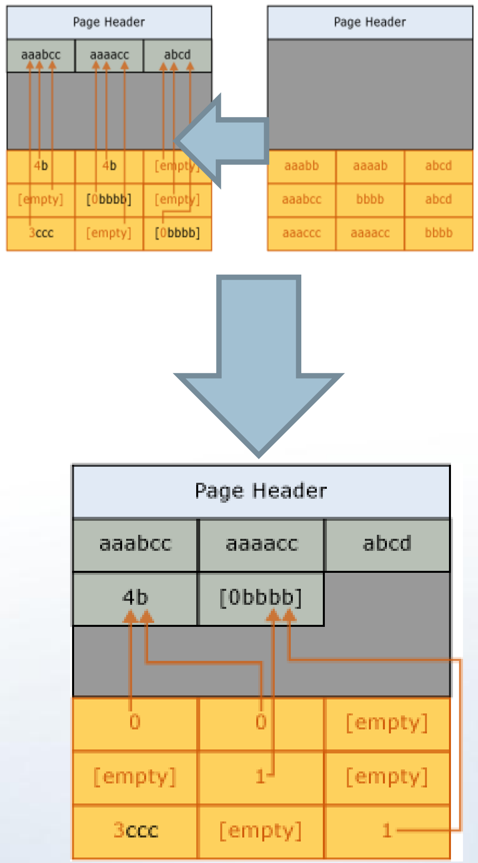
*at the byte level, page compression is data type agnostic

Page Compression Implementation

- Does the following:
  1. Row compression
  2. Prefix compression
  3. **Dictionary compression**
  4. Unicode Compression

Dictionary Compression:

- Searches for repeated values anywhere on the page, and stores them in the CI area
- Is not restricted to one column as per prefix



CI = compression information structure

Page Compression Implementation

- Does the following:
  1. Row compression
  2. Prefix compression
  3. Dictionary compression
  4. **Unicode Compression**

Unicode Compression:

- Works on Row and Page compressed data
- Reduces length of datatype storage due to under-utilised storage (double byte) capability.
- NCHAR & NVARCHAR Data can see savings of up to 50%*
- Does not work for off row or in nvarchar(max)

* 50% for data that is wholly single byte

Demo

Identifying what would be good for compression.

Experiment

Sparse Columns

- A type of column storage where a NULL takes no storage at all
  - normal NULL columns require space to indicate that they are NULL
- The data is stored internally in a different form
- The Keyword SPARSE is used in the column creation definition
- Sparse columns can also be used as a column set, which then allows you to return data that is only in sparse columns
- Best used when you have many columns that are mostly going to be empty/NULL, otherwise slightly inefficient.

Demo

A look into how to implement & digest sparse columns

Experiment

Filtered Indexes

- An index with a WHERE clause!
- Can be used with Index Intersection etc in the Query Optimiser
- Allows statistics to be more focused as the dataset within the index is distilled
- Compression Capable
- Overhead is in the Administration

Demo

Simple example of efficiency introduced through a Filtered Index

Experiment

Clustered ColumnStore Indexes

- Works on individual columns rather than rows
- Read-ahead for Columnstore can be up to 8MB!
- Three basic phases for Columnstore Index creation:
  - Row Group Separation
  - Segment Creation
  - Compression

- Types of ColumnStore Indexes
  - Clustered (Updateable)
  - Non Clustered (not updateble)

- Opton Also Available:
  - ColumnStore_Archive

ColumnStore Indexes

According to SQL MVP Niko Neugebauer

- Compression techniques applied to achieve super compression are the following:
  - Value Scale
  - Bit Array
  - Run-length Encoding
  - Dictionary encoding
  - Huffman encoding
  - Lempel-Ziv-Welch
  - Binary compression

Links and Resources

- Storing Decimal Data As Variable Length
  - Mark Rasmussen
- The Anatomy of Vardecimals
- SQL Server Diagnostic Information Queries
  - Glenn Berry
- Pro SQL Server 2012 Relational Database Design and Implementation
- Microsoft SQL Server 2008 Internals
- SQL Server 2014 : Native backup encryption & compression
  - Aaron Bertrand
- Columstore Indexes : Niko Neugebauer

Thank you!