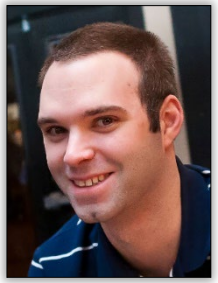
The Ins and Outs of SQL Server Data Compression

Bob Pusateri

About Bob Pusateri



Microsoft
CERTIFIED
Master

SQL Server 2008

@sqlbob
bob@bobpusateri.com
bobpusateri.com
bobpusateri

MVP
Microsoft®
Most Valuable
Professional

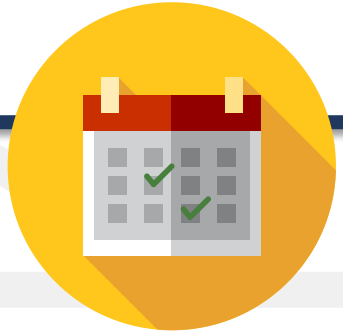
Specialties:

- Performance Tuning
- Very Large Databases
- SQL Server Storage Engine
- Cloud Architecture
- Big Data
- Teaching/Mentoring/Helping

ChiSQL

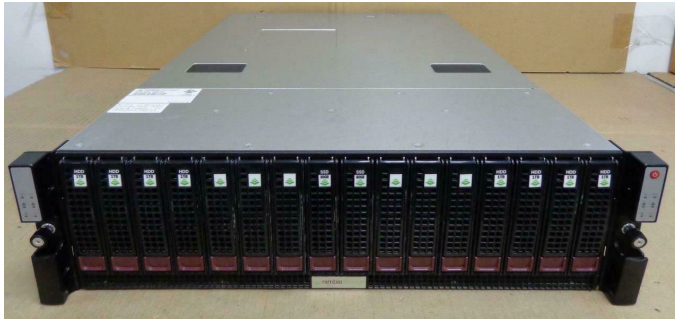
FRIEND OF
redgate
2021

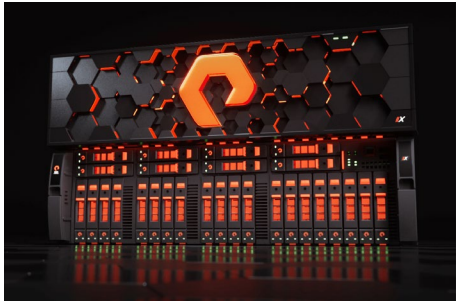
Agenda



- Compression Basics/Primer
- Row & Page Compression
- Columnstore Compression
- Questions

Why Compress?

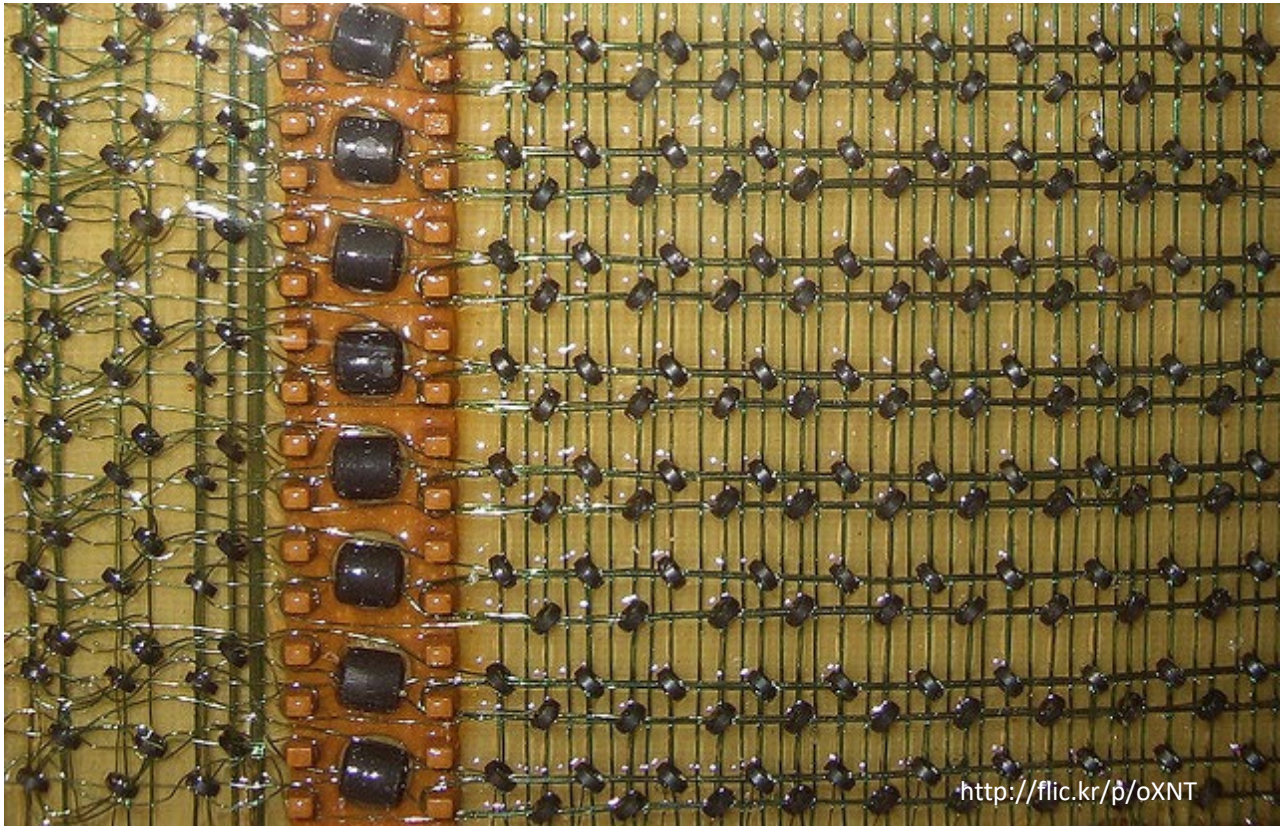




Why Compress?

http://flic.kr/p/6tGK8E

Why Compress?



Why Compress?

Lossy Compression

Lossless Compression

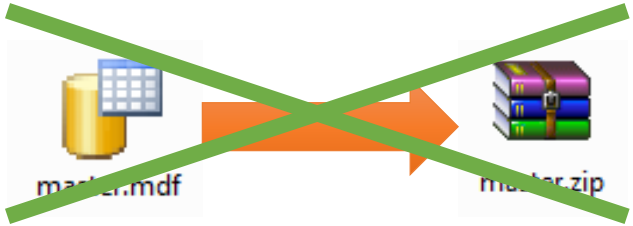
http://flic.kr/p/5BbUo2

Lossless Compression

- Huffman Coding
  - Most common data coded with fewest number of bytes
- Statistical Modeling
  - LZ77/LZ78
  - LZW/LZMA/gzip
  - Create a dictionary of most common patterns
  - Reference them with pointers where appropriate

SQL Server Access Patterns

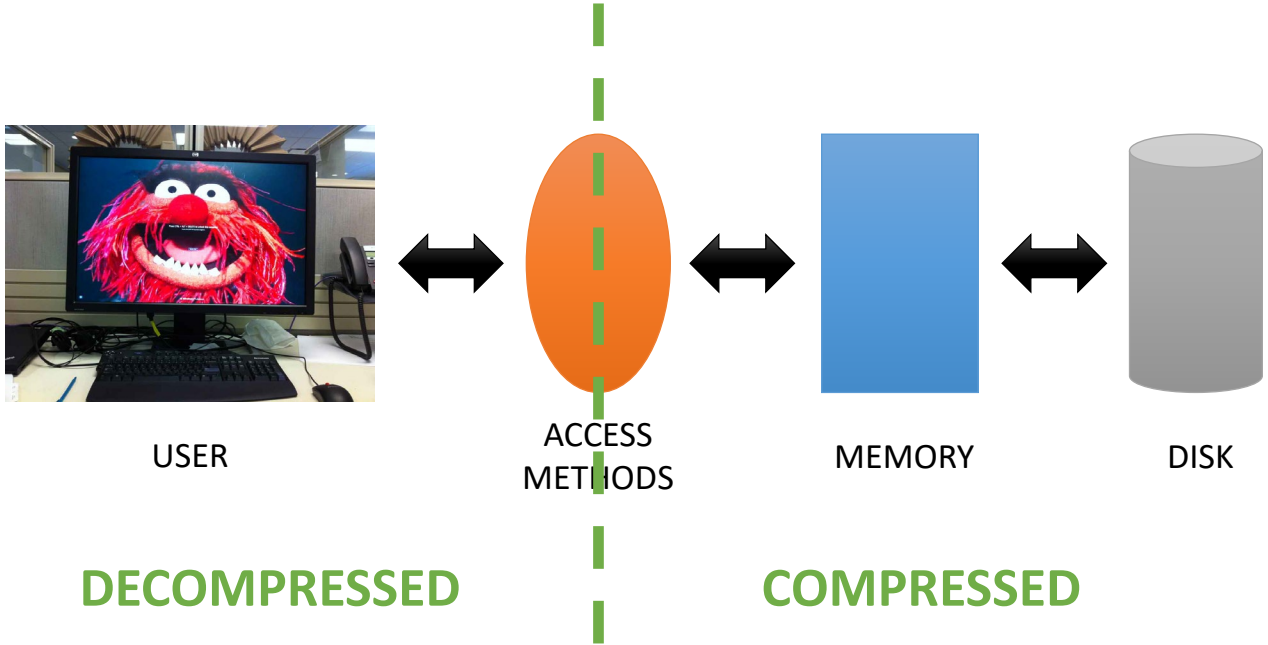
- Why not just zip* the .mdf file?



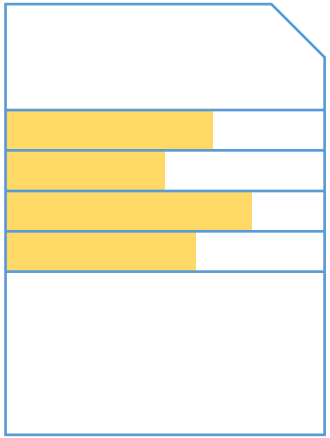
Database I/O is:

- Generally very random
- Generally very small (8KB pages, 64KB extents)
- NEED TO BALANCE COMPRESSION RATIO & SPEED!

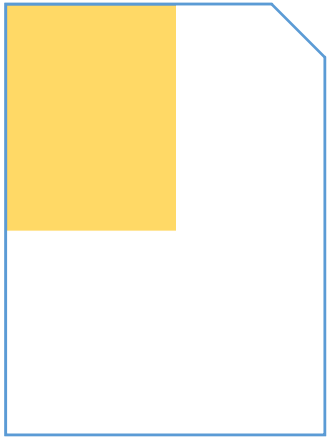
Where Data Compression Occurs



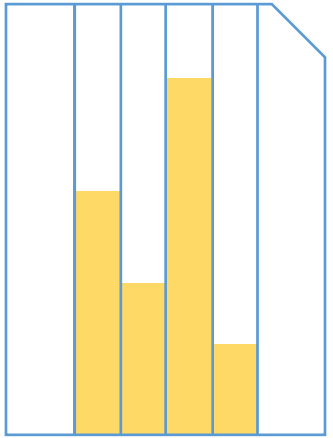
Compression Types



ROW



PAGE



COLUMNSTORE

Row Compression

ROW Compression

- “Only use the space you need” compression
- Fixed-length numeric data becomes variable-length
  - Only uses necessary bytes
  - INT with value of 5 will use 1 byte instead of 4
- Fixed-length character data also variable-length
  - CHAR(100) value ‘Stu’ uses 3 bytes, not 100
- NULL & zero values take no space
- Variable-length metadata overhead reduced
  - New record structure
  - 2 bytes per column without compression
  - 4 bits per column with compression
  - For variable-length columns <= 8 bytes
  - “Column Descriptors”

http://flic.kr/p/7f7Nb9

ROW Compression

- Has no effect on smallest datatypes
  - TINYINT, SMALLDATETIME, DATE, TIME
- Does help BIT columns
  - 4 columns can be shrunk to 4 bits (instead of 8/8)
- Unicode character data (nchar/nvarchar) MAY BE compressed
  - Uses SCSU algorithm
  - Only if compression has an advantage
  - nvarchar(max) not compressed at all

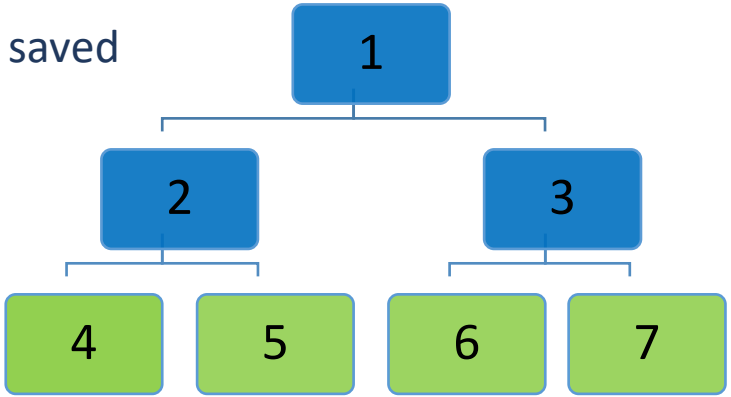
ROW Compression

ID SMALLINT		FirstName CHAR(8)							
92		S	A	M					
4346		N	I	C	K				
120		M	I	C	H	E	L	L	E

Page Compression

PAGE Compression

- Has 3 components (applied in this order)
  - ROW compression
  - Column prefix compression
  - Column dictionary compression
- Adds a “CI” (compression information) structure to each page
- May not be applied to every page
  - Only utilized on pages where space can be saved
- Applies only to leaf-level pages

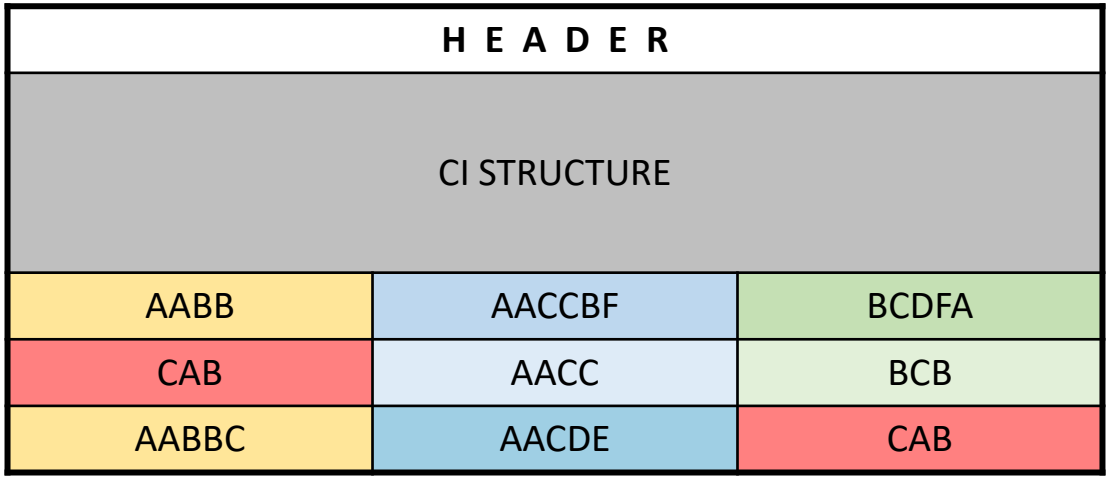


PAGE Compression (prefix)

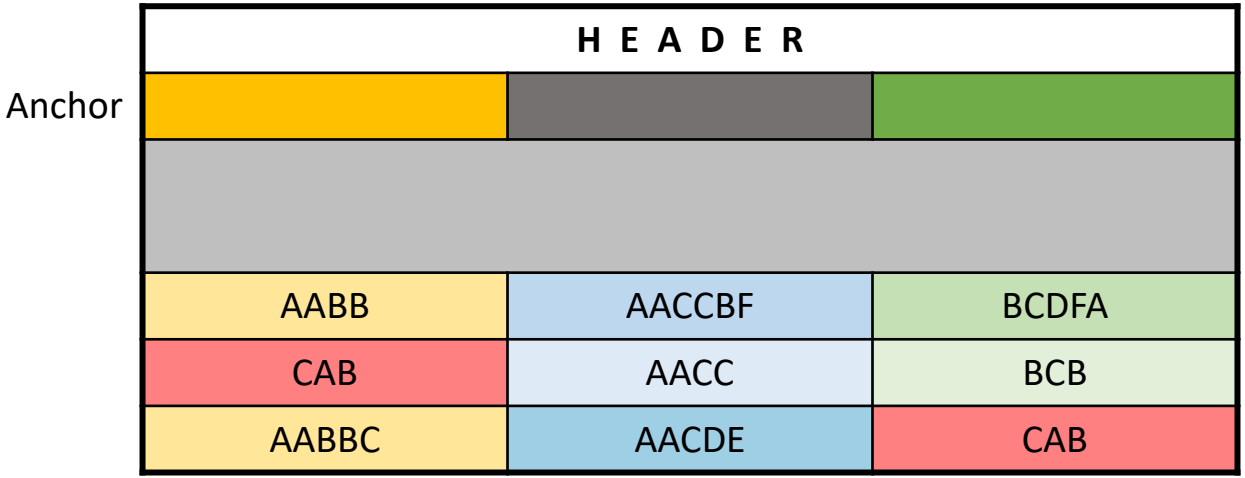
- Prefix is in terms of byte values (type agnostic)
- Largest, most common prefix stored in “anchor record” in CI structure
- Row values are stored in terms of the anchor value

```
0x012368FFB8
0x012367FFB8
```

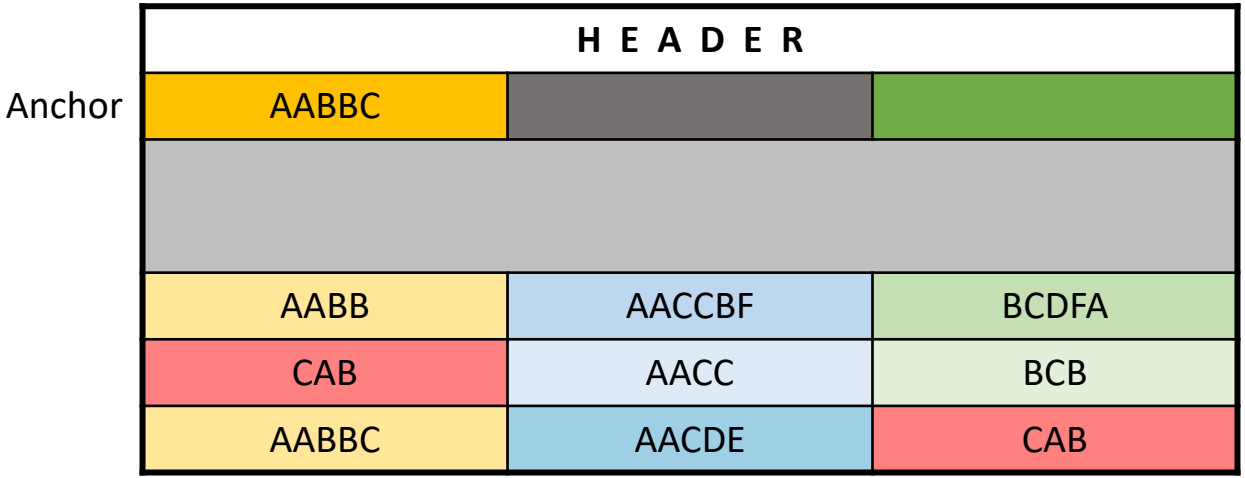
PAGE Compression (prefix)



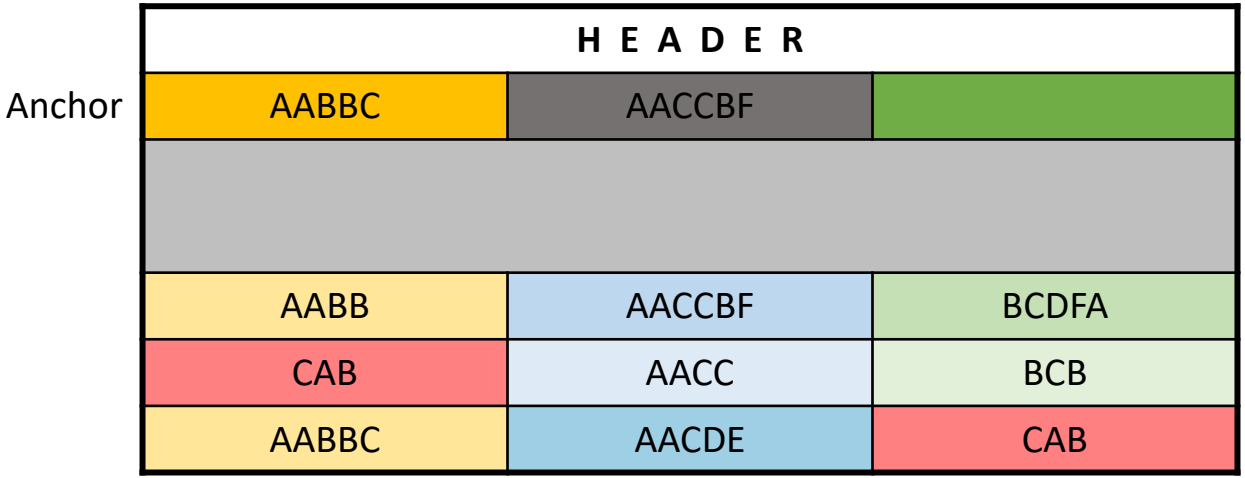
PAGE Compression (prefix)



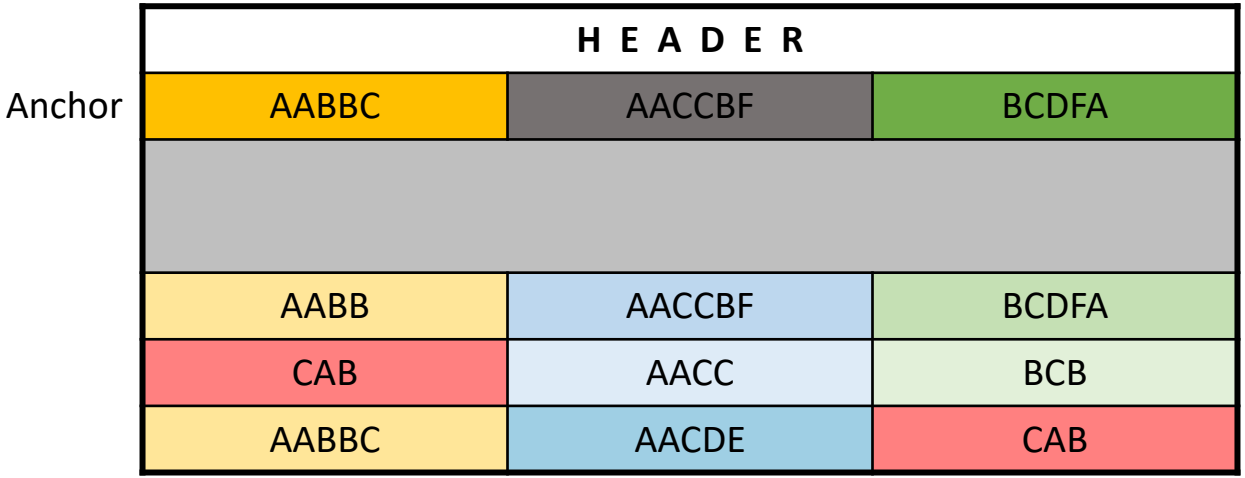
PAGE Compression (prefix)



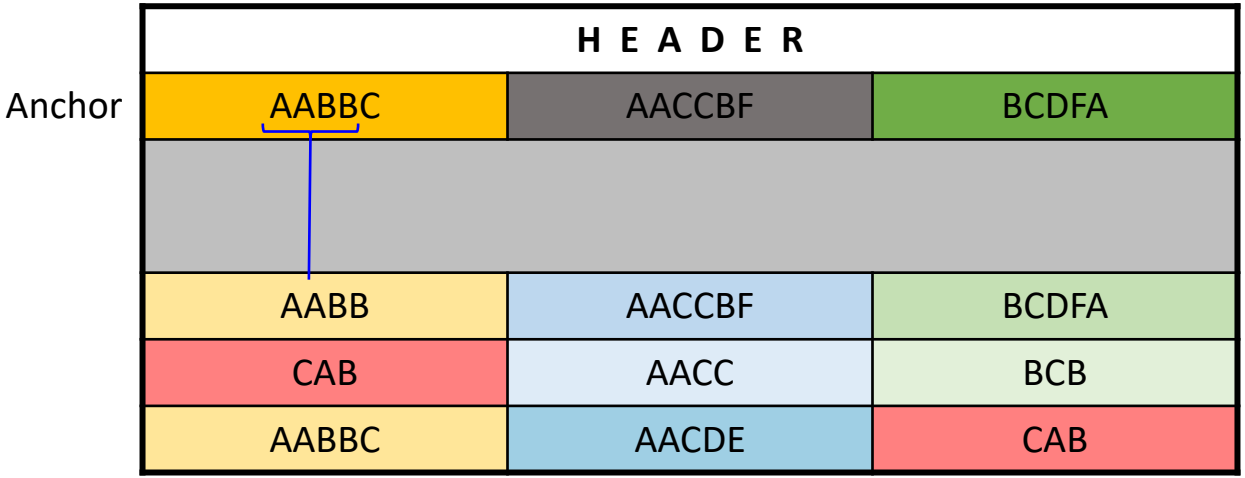
PAGE Compression (prefix)



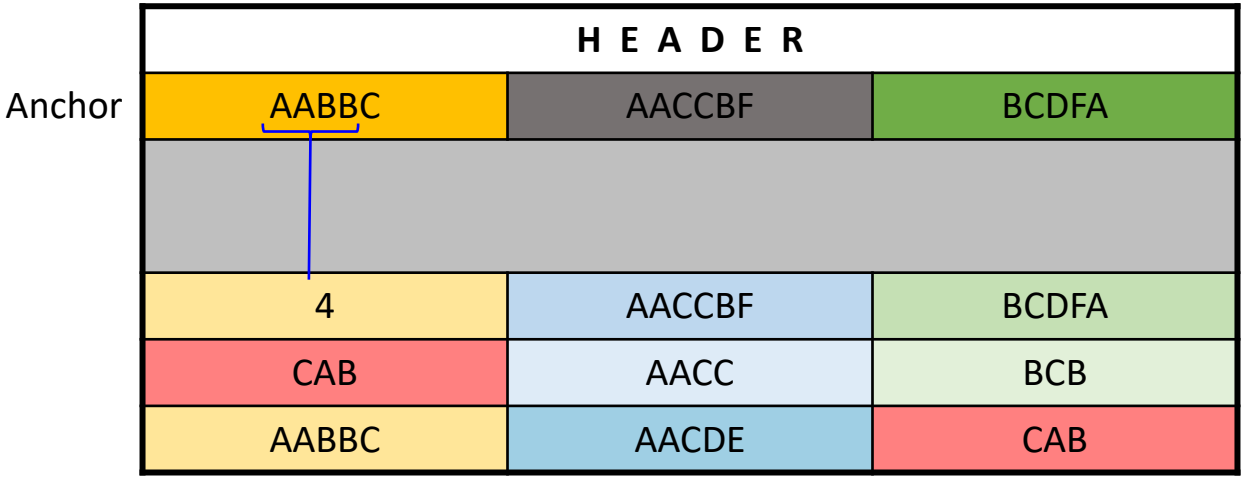
PAGE Compression (prefix)



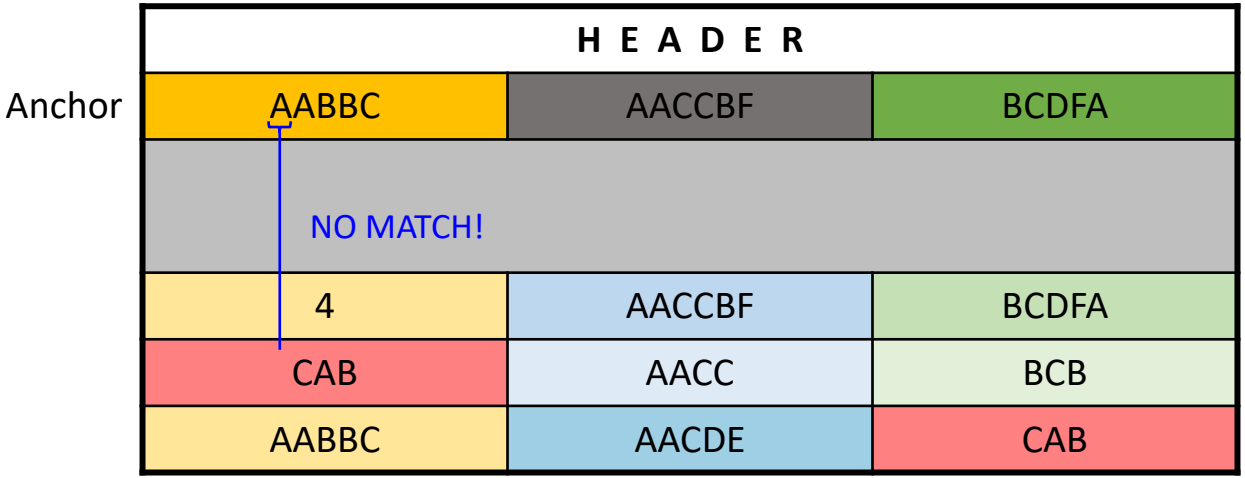
PAGE Compression (prefix)



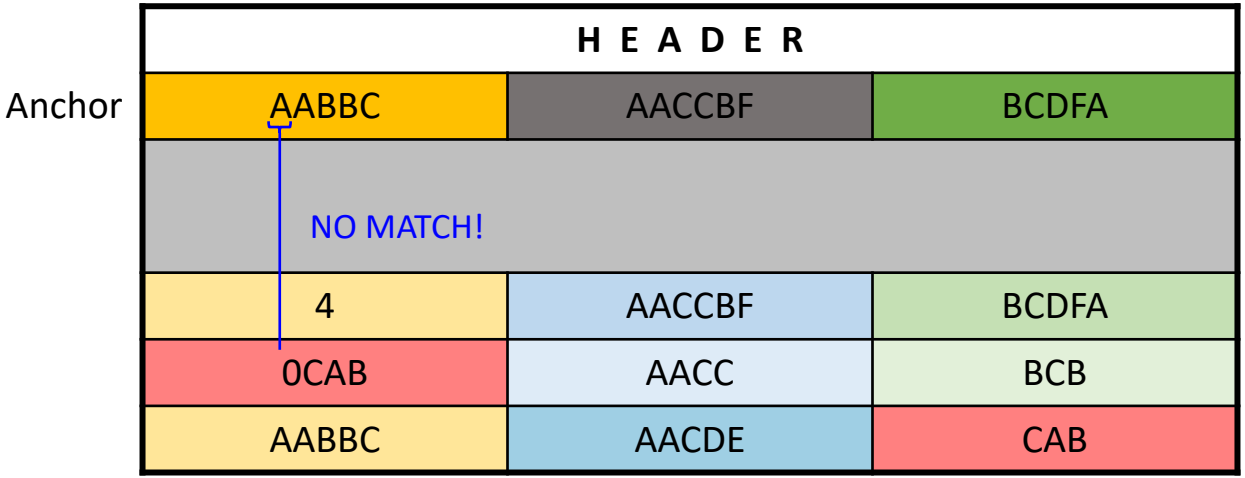
PAGE Compression (prefix)



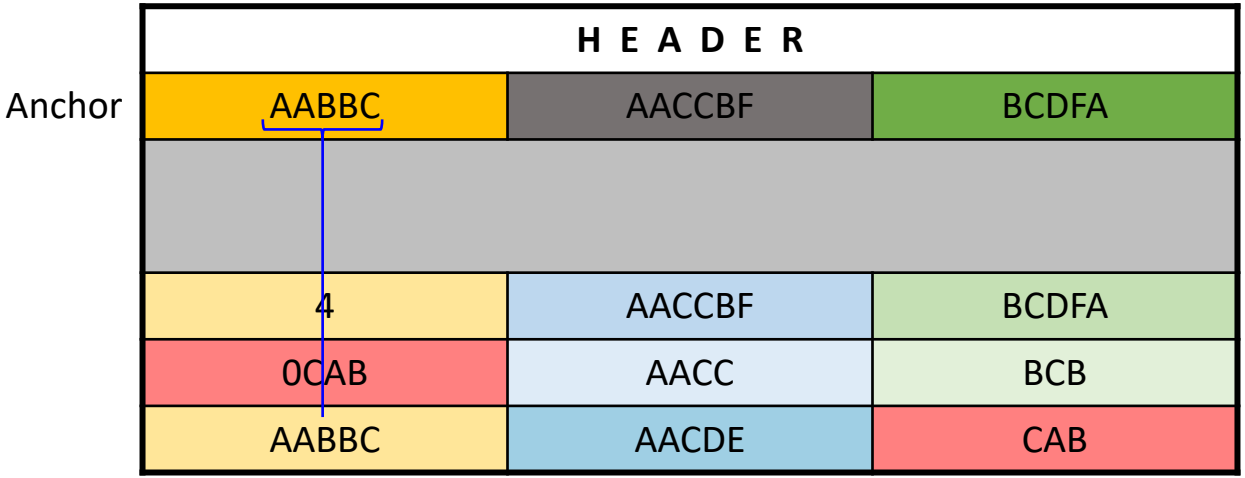
PAGE Compression (prefix)



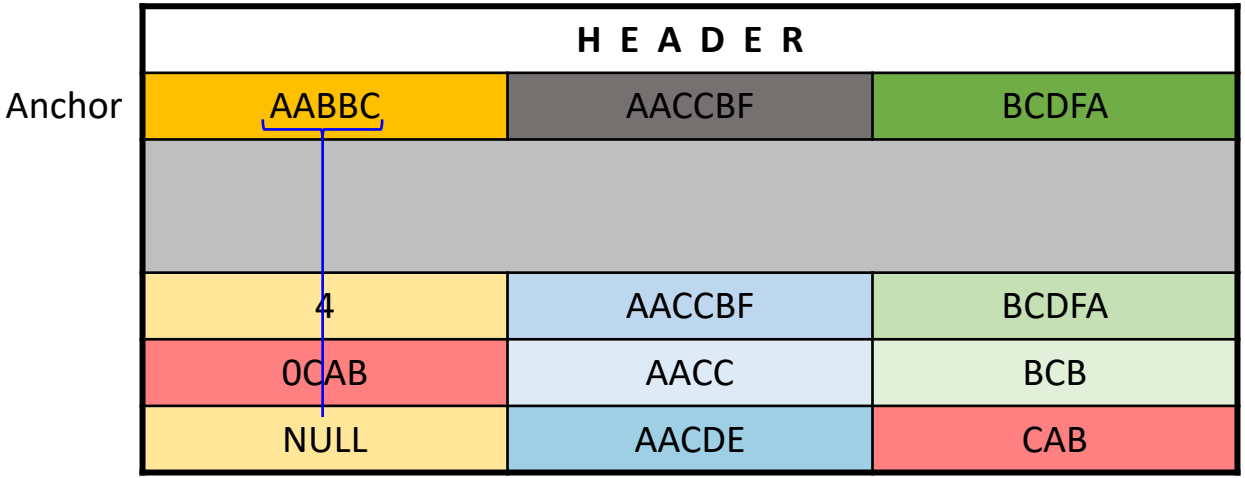
PAGE Compression (prefix)



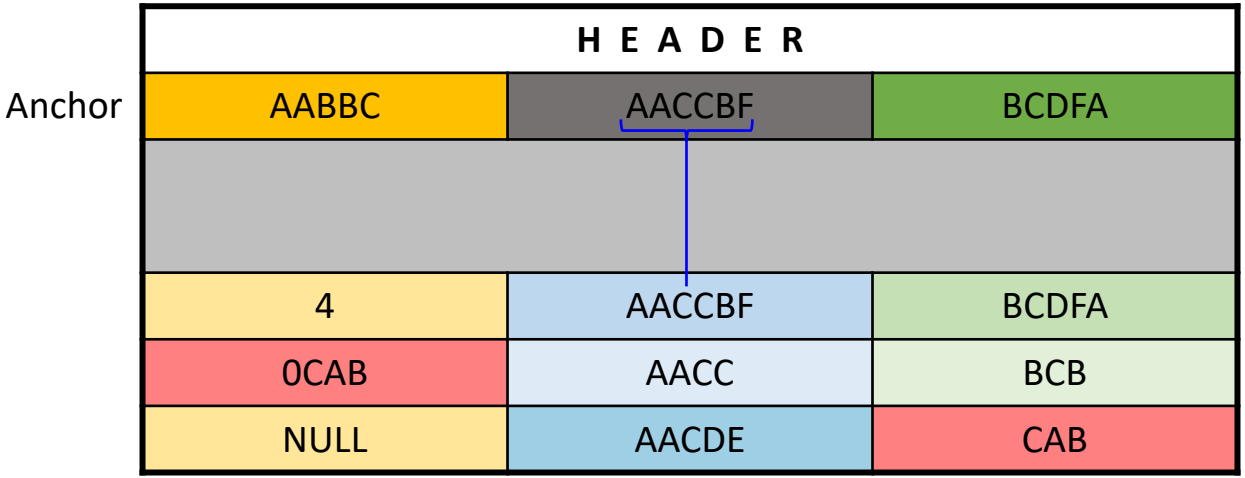
PAGE Compression (prefix)



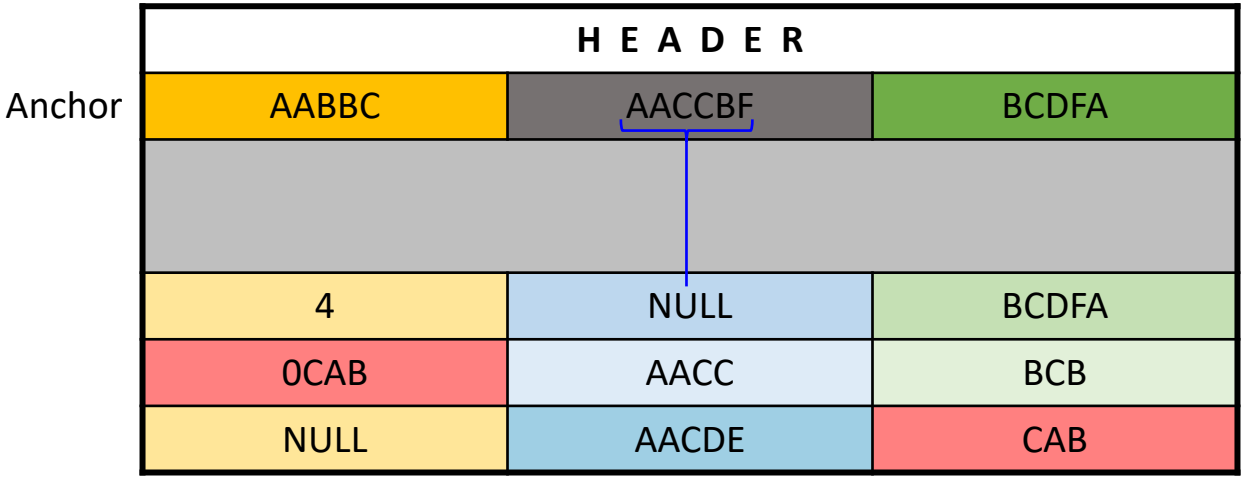
PAGE Compression (prefix)



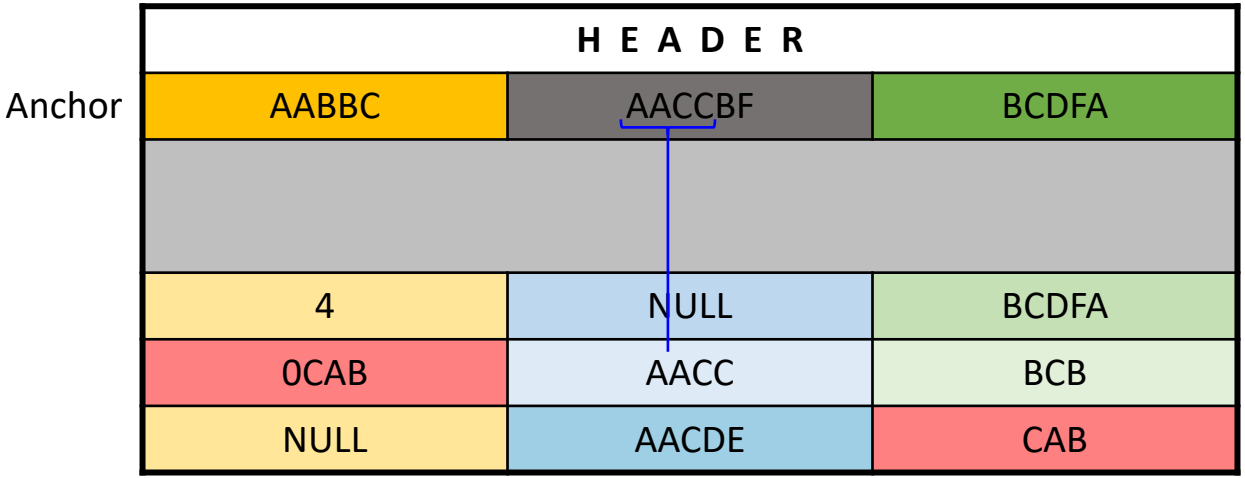
PAGE Compression (prefix)



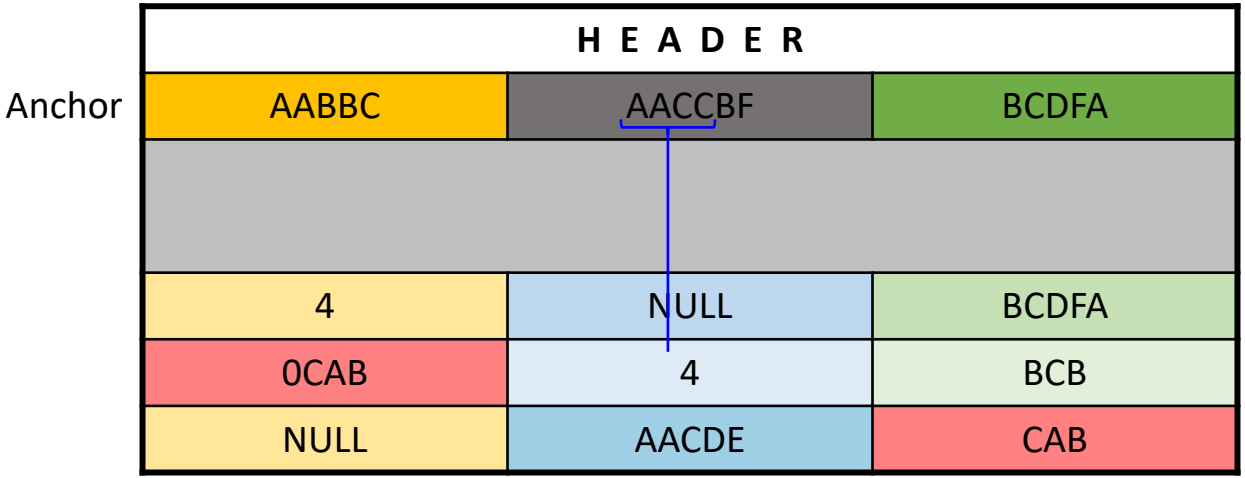
PAGE Compression (prefix)



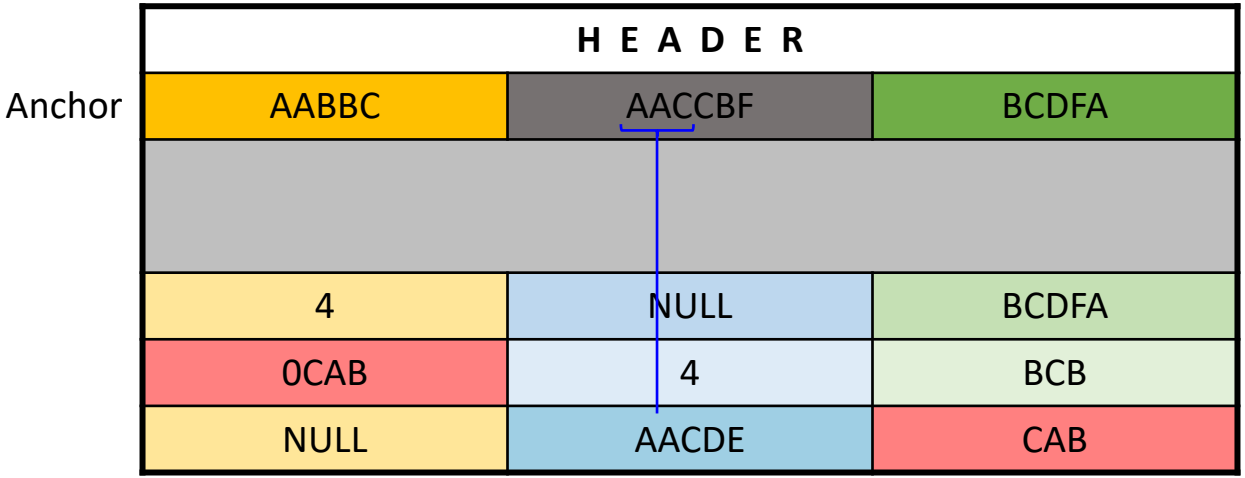
PAGE Compression (prefix)



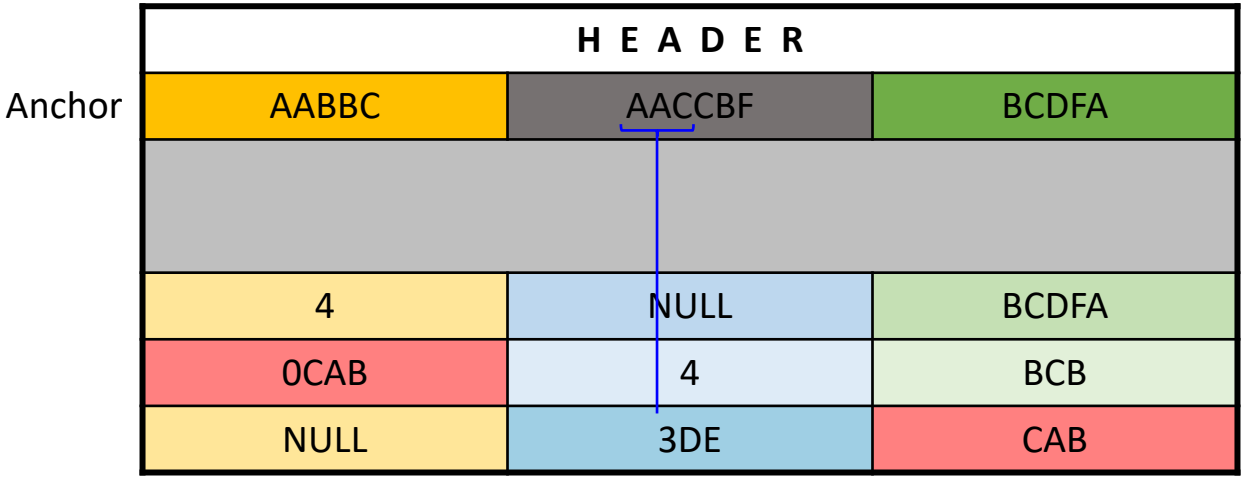
PAGE Compression (prefix)



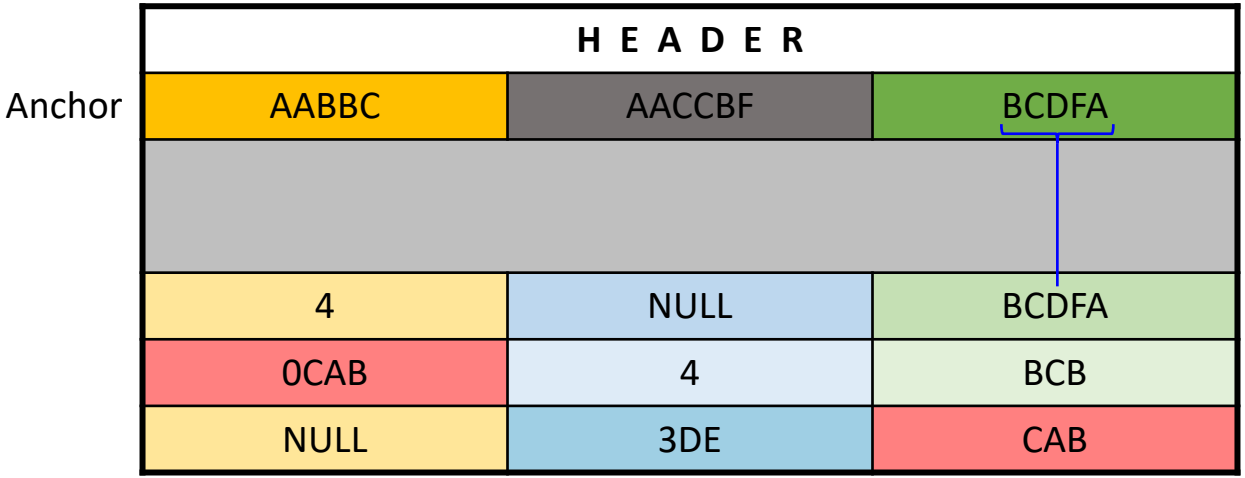
PAGE Compression (prefix)



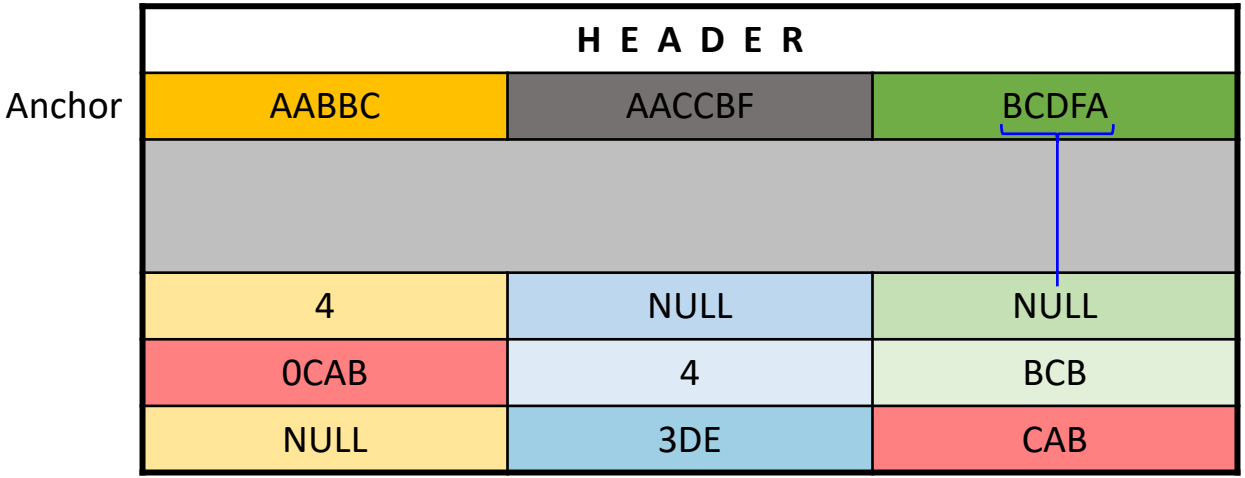
PAGE Compression (prefix)



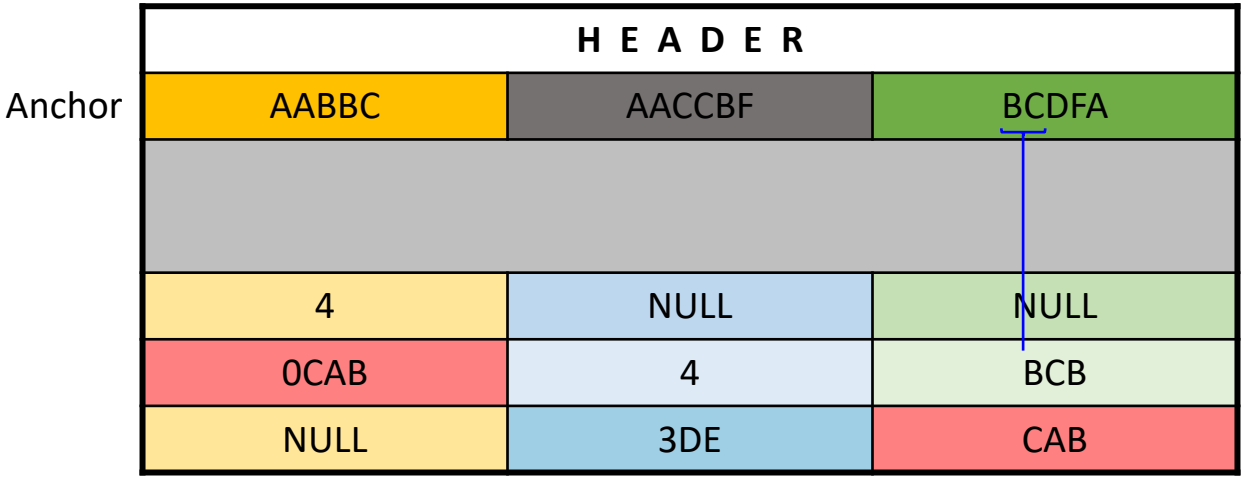
PAGE Compression (prefix)



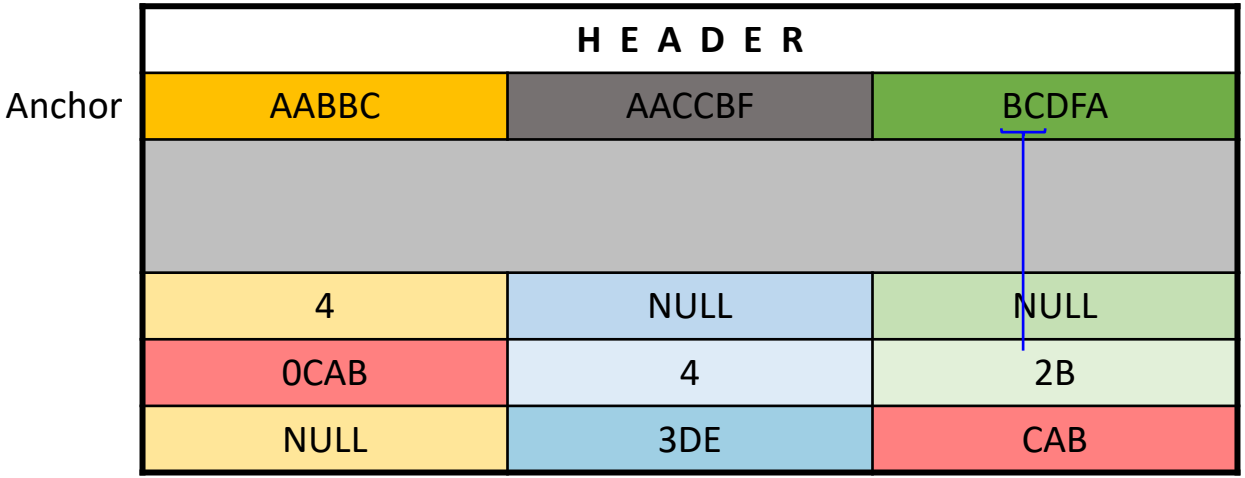
PAGE Compression (prefix)



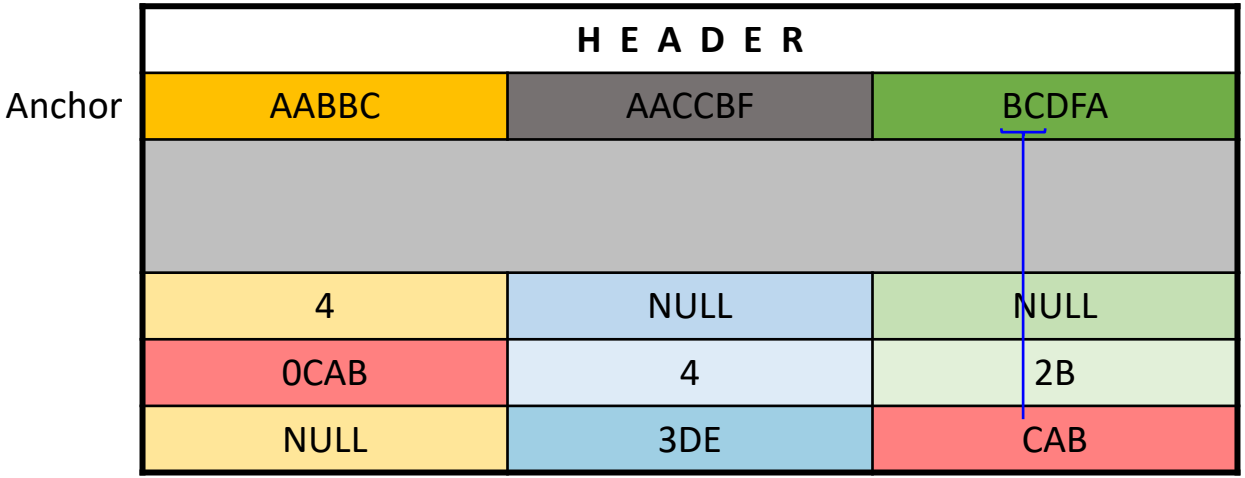
PAGE Compression (prefix)



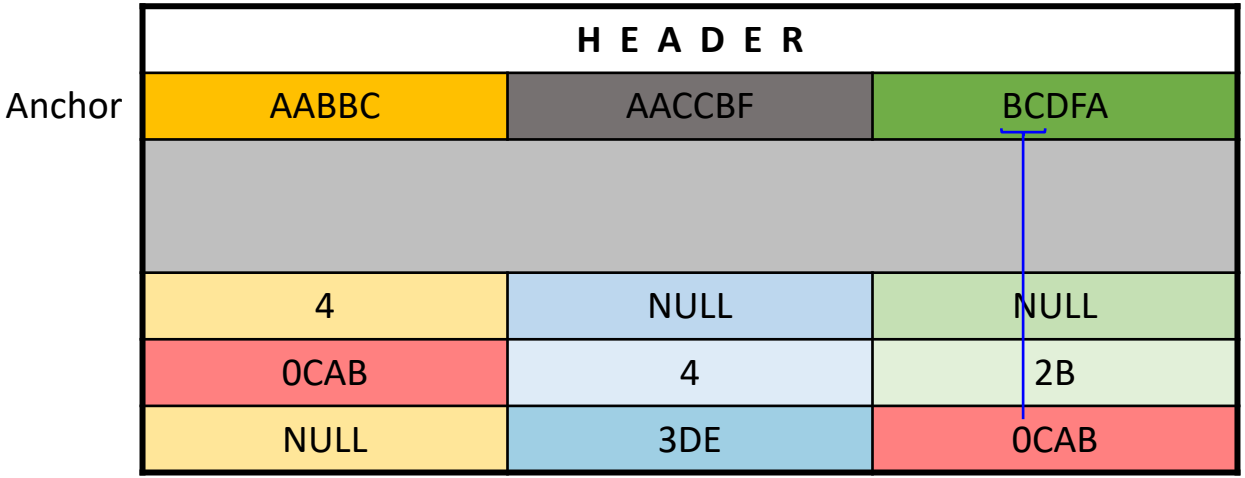
PAGE Compression (prefix)



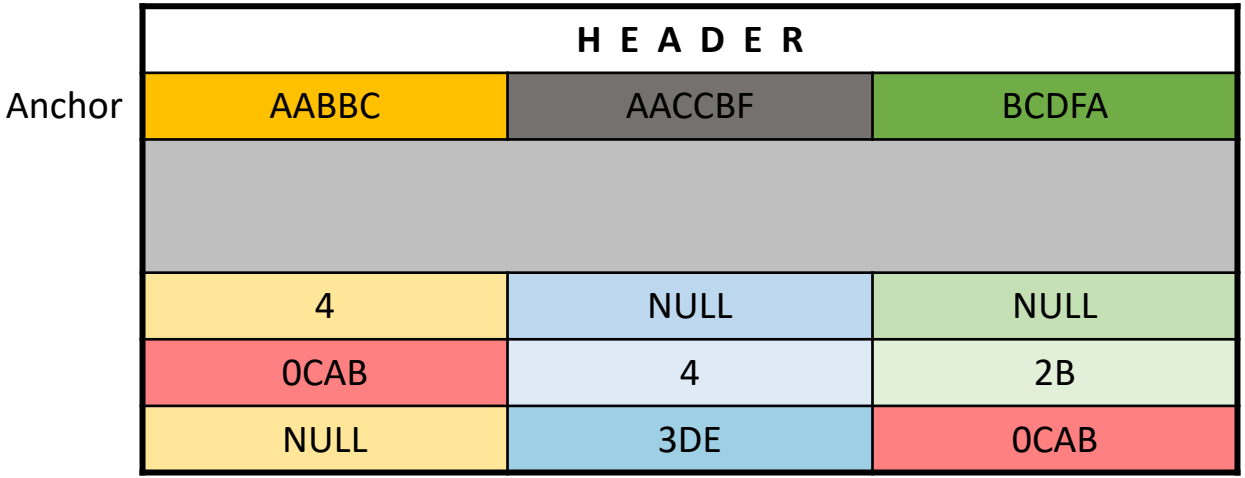
PAGE Compression (prefix)



PAGE Compression (prefix)

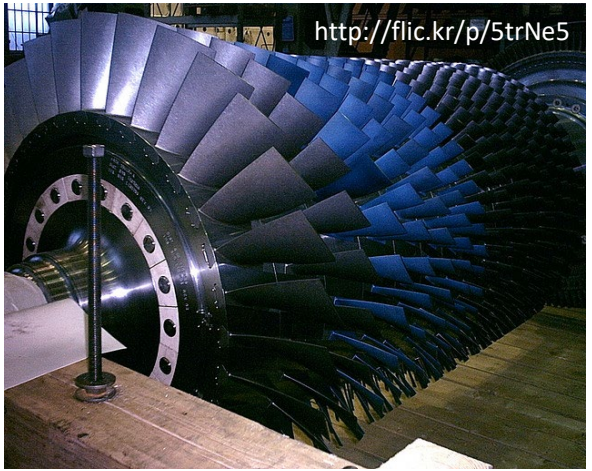


PAGE Compression (prefix)

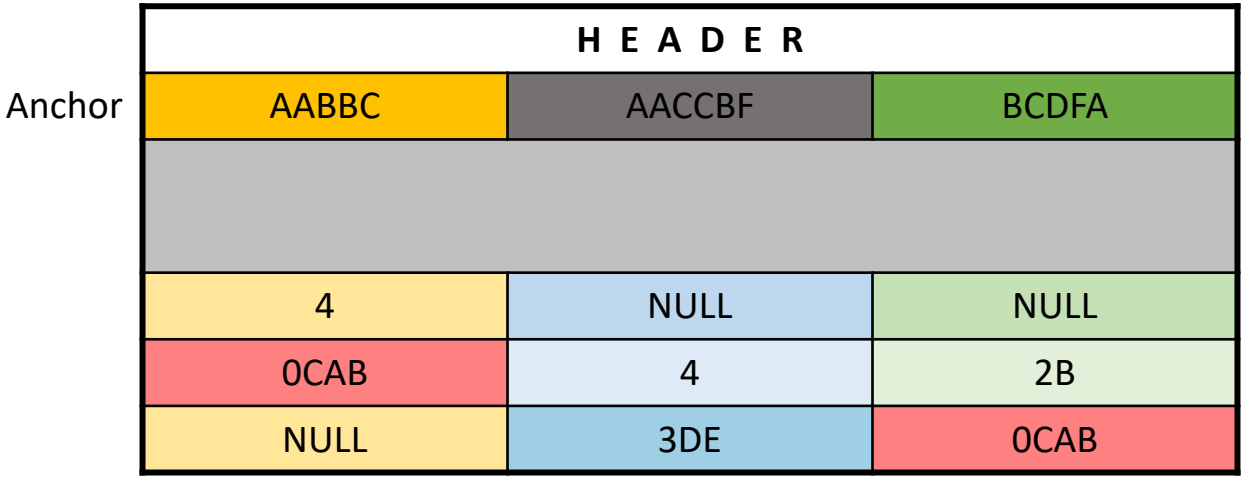


PAGE Compression (dictionary)

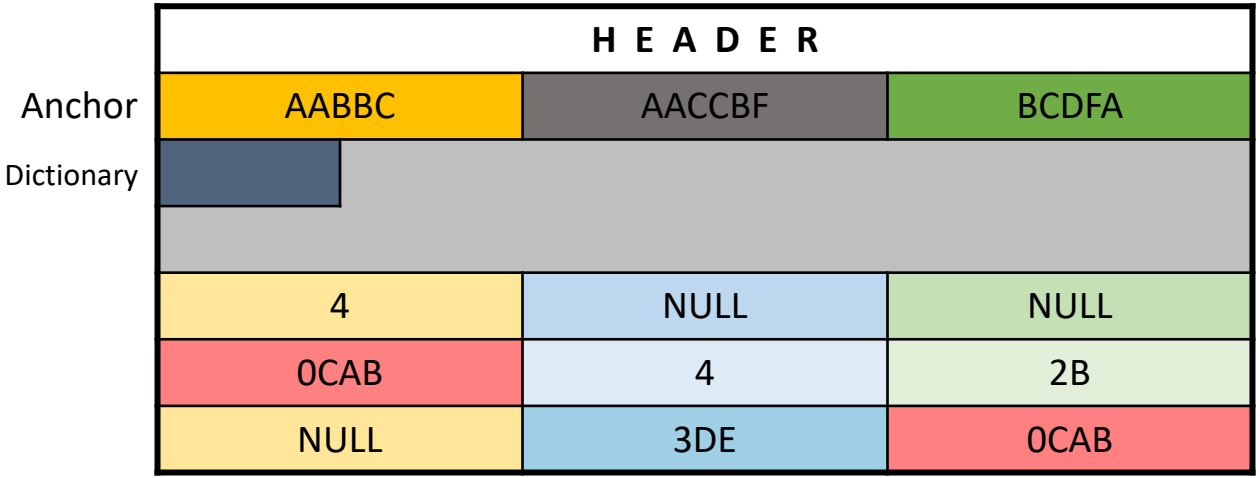
- De-duplicates common byte values over ALL columns on page
- Dictionary entries stored as a 0-based array
- Array is stored after anchor row in the CI structure
- Entries created only if a value repeats in 2 or more columns



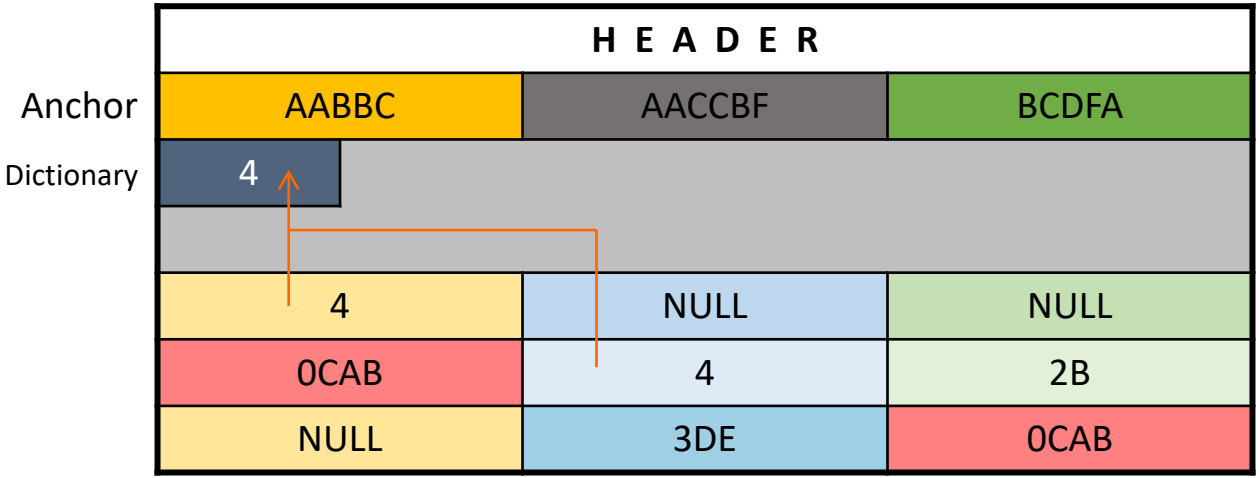
PAGE Compression (dictionary)



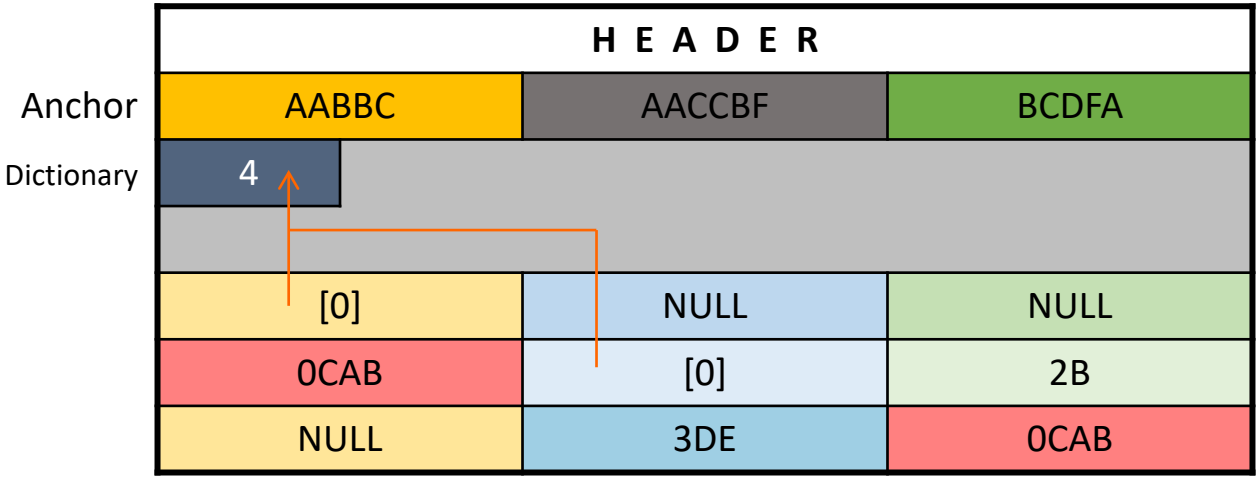
PAGE Compression (dictionary)



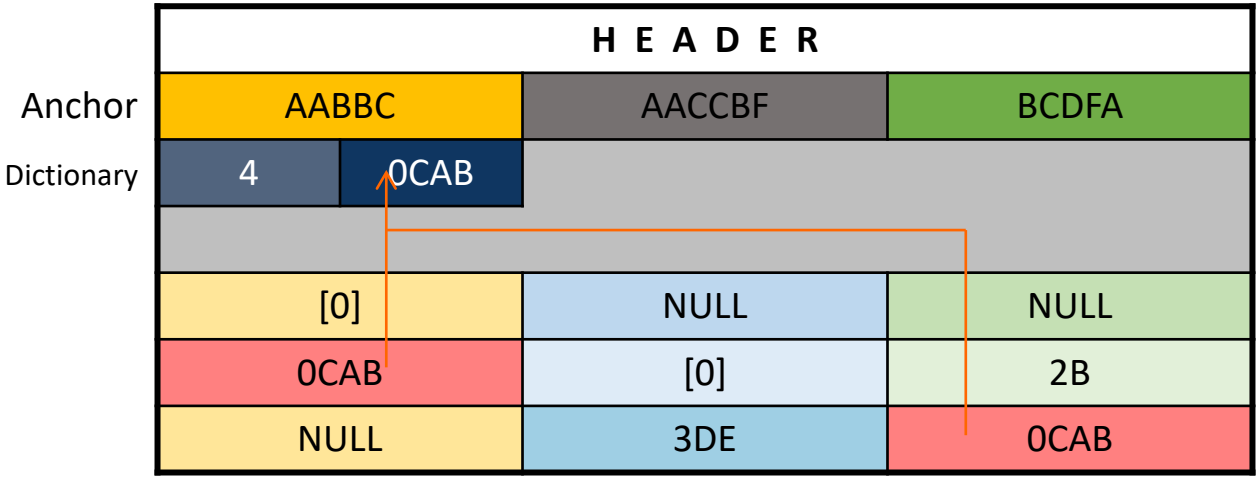
PAGE Compression (dictionary)



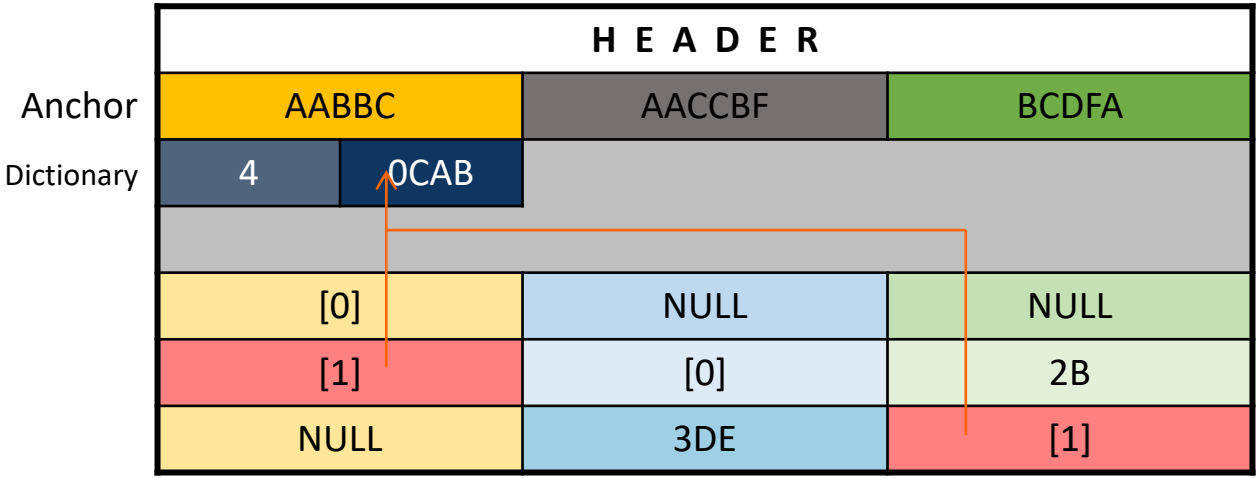
PAGE Compression (dictionary)



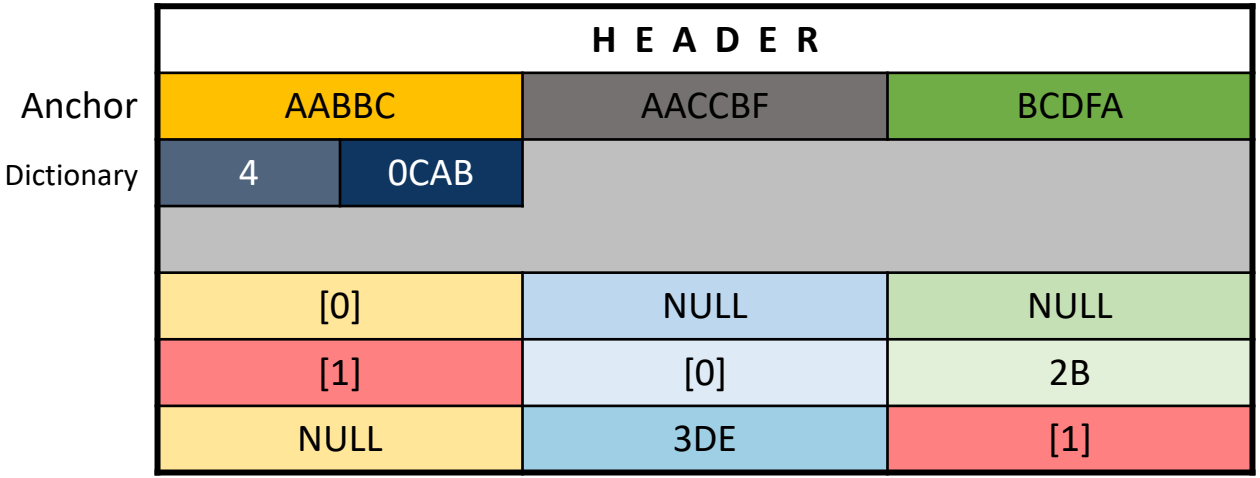
PAGE Compression (dictionary)



PAGE Compression (dictionary)



PAGE Compression (dictionary)

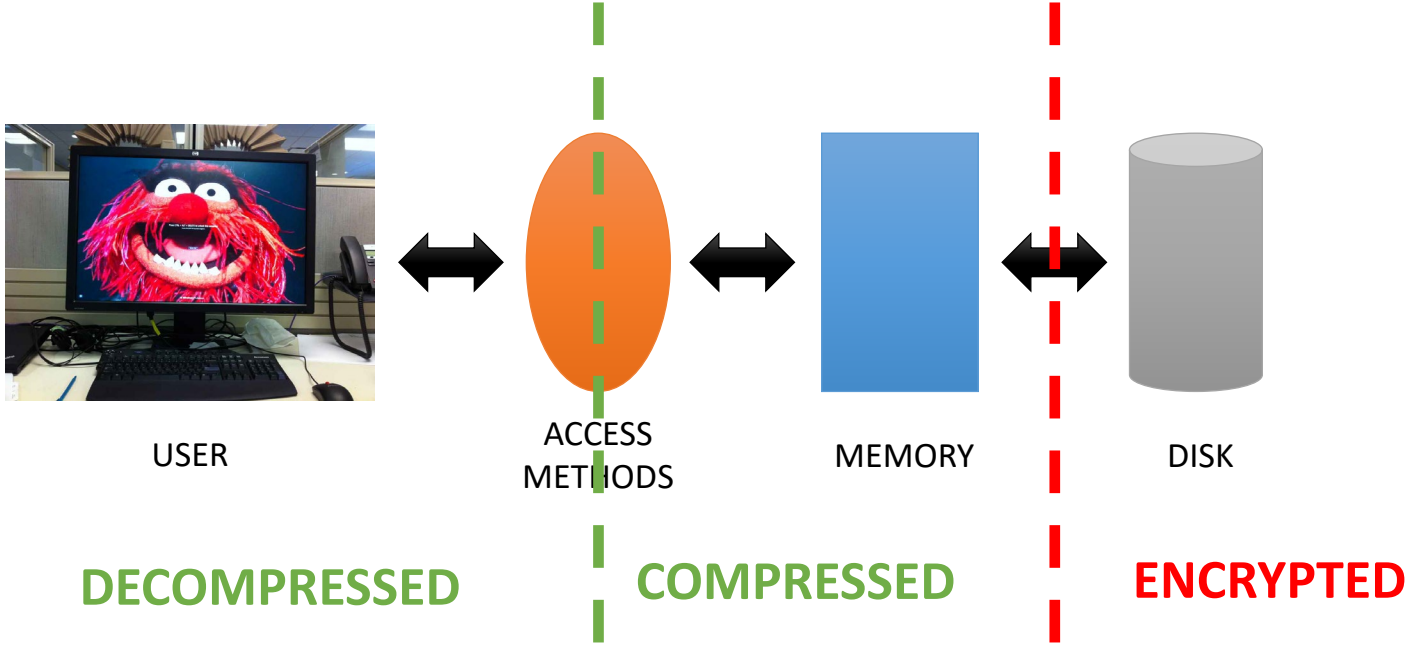


PAGE Compression

- CI Structure is rebuilt on an all-or-nothing basis
  - Updating 1 row probably won't trigger a rebuild
- Page modifications tracked in the CI Structure

- If PageModCount >25
- OR PageModCount / Rows on Page > 25%
- Then **TRIES** to rebuild CI Structure
- (Rebuild is only applied if rebuild creates enough space for 5 or 25% more rows)

Data Compression and TDE



Backup Compression is Different

- Data compression uses the described methods to shrink data in hopes of fitting more records on a page
- Backup compression uses a proprietary algorithm to compress the backup stream in hopes of making the backup file(s) smaller
- Both methods will use more CPU in hopes of reducing IO
- It is possible to use both methods together
  - Net effect of backup compression will be lower if data compression is already in place

What Can't Be Compressed?

- System Tables
- Anything not on a data or index page
  - Binary/LOB types (VARCHAR, TEXT, NTEXT, VARBINARY, IMAGE, SQL_VARIANT, UNIQUEIDENTIFIER, XML)
  - FILESTREAM
  - NVARCHAR(MAX) even when stored in-row

How Much Space Is Saved?

- It Depends!
- Proper datatyping reduces compression ratio
- Fewer repeating values also reduces compression ratio

- `sp_estimate_data_compression_savings`
  - Try before you compress!
  - Samples 5% of object in TempDB
  - Compresses & extrapolates

Data Compression Compatibility

- Introduced in SQL Server 2008
- WAS Enterprise Edition only until 2016 SP1
  - Would only restore on Enterprise Edition!
  - Didn't tell you this until the end of a restore!!
  - `sys.dm_db_persisted_sku_features`
- Can't be used with Sparse Columns

When To Use Row/Page Compression

- Most effective on tables with high number of scans
  - Savings are smaller from single lookups (reads or writes)
- Tables that have a high percentage of reads / READ_ONLY
- Data that is seldom accessed
- All the time?

What If My Storage Array Does Compression?

- Test it!
- Pros
  - Will probably compress better than Row/Page Compression
  - Not using database server CPUs to do this work
- Cons
  - Data won't be compressed in memory

Columnstore Compression

Columnstore over time

- Introduced in SQL Server 2012
  - Was read-only, nonclustered indexes only
  - Enterprise Edition only
- 2014
  - Added updateable clustered columnstore indexes
  - Still EE-only
- 2016
  - Updateable nonclustered columnstore indexes
  - Secondary b-tree indexes
  - Available in standard edition! (needs SP1)

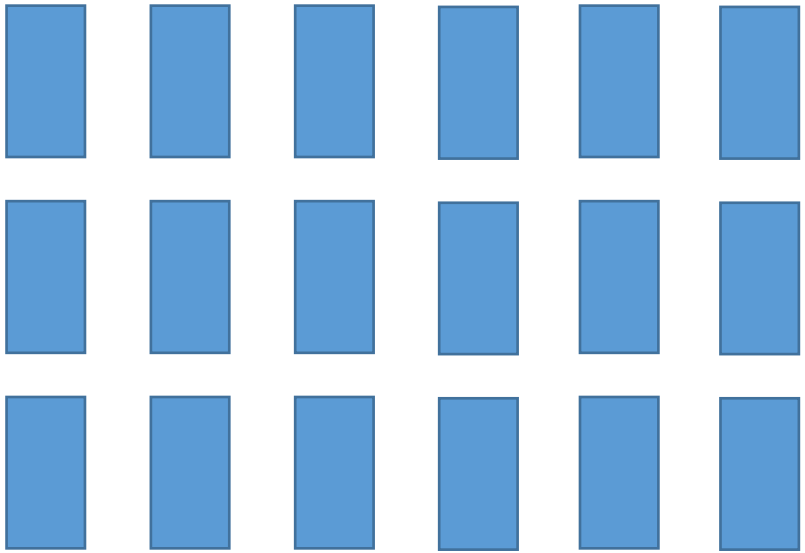
Columnstore over time

- 2017
  - Adaptive Query Processing added for tables w/ columnstore indexes
  - Can add computed columns to clustered columnstore indexes
- 2019
  - Online build/rebuild of clustered columnstore indexes

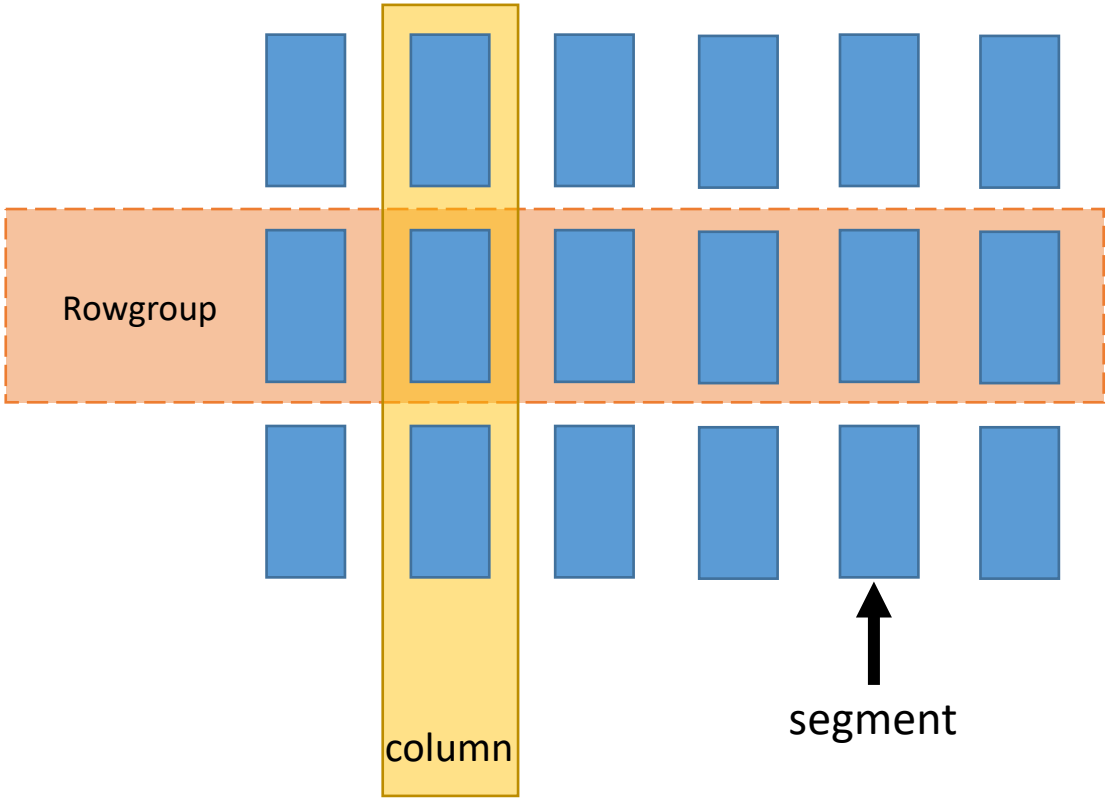
Columnstore

Relational Table (Rowstore)

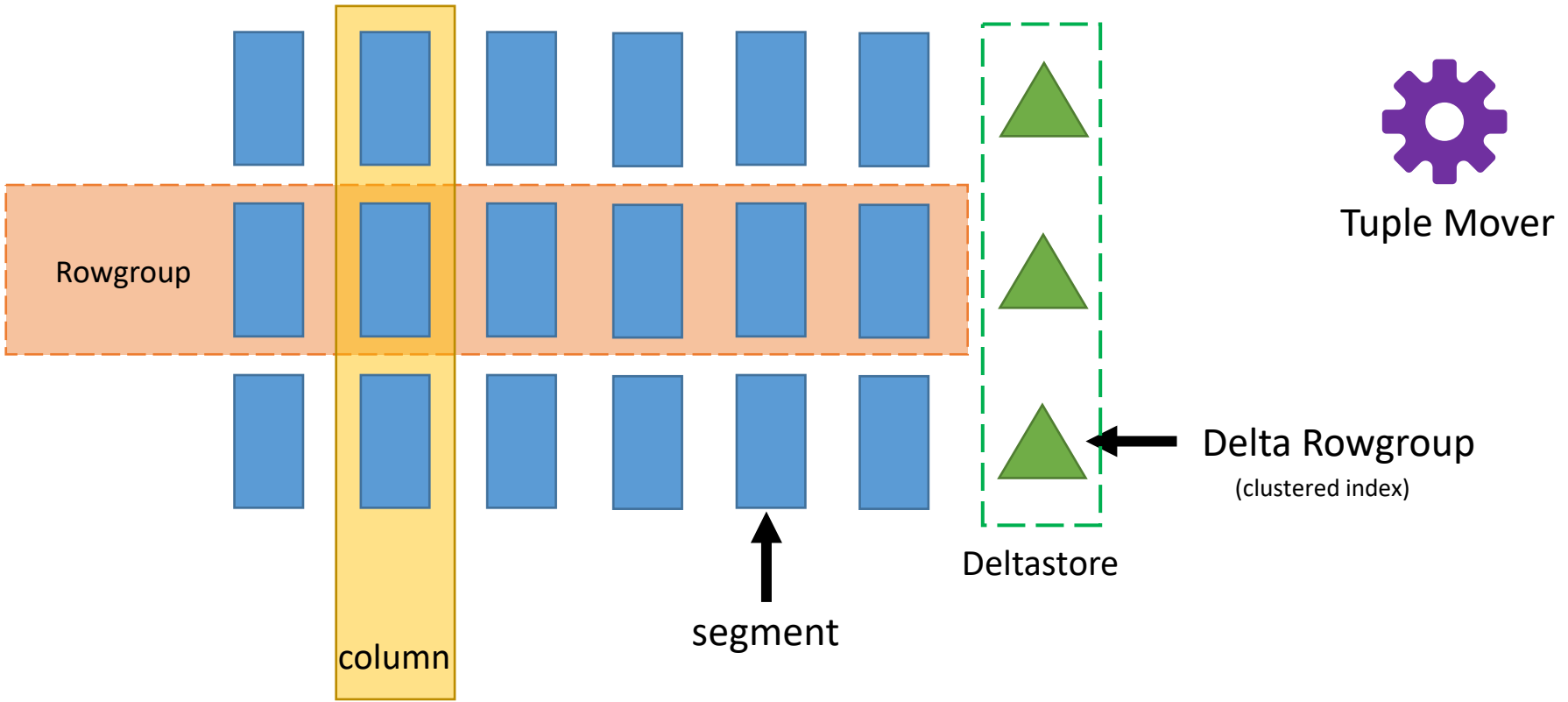
Columnstore



Columnstore



Columnstore



Columnstore Compression Algorithms

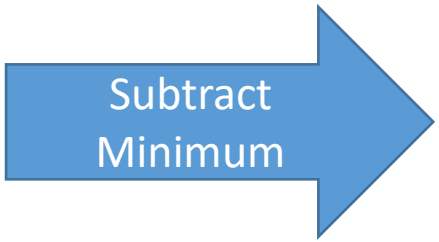
- Several compression algorithms possible
  - Decided upon by the engine based on profile of the data
  - You get no control over this

Value-Based Encoding (aka “Value Scale”)

- Create a mathematical relationship between value & stored value

Value
1105
1109
1122
1113
1107

Min: 1105
Max 1122
11 bits needed



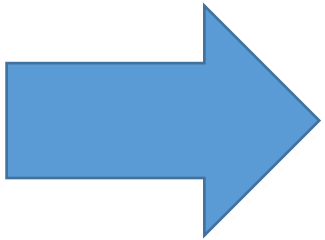
Value
0
4
17
8
2

Min: 0
Max 17
5 bits needed
“base” = 1105

Bit Array

- Stores a bit column for each distinct value found

Blood Group
A
O
AB
O
A
B
A

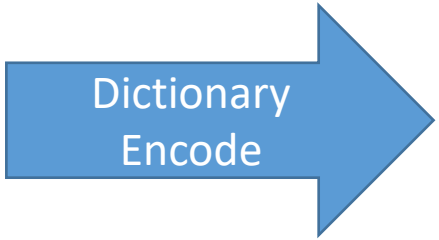


A	B	O	AB
1	0	0	0
0	0	1	0
0	0	0	1
0	0	1	0
1	0	0	0
0	1	0	0
1	0	0	0

Dictionary Encoding

- Build a dictionary of distinct values & replace data with them

Vehicle Make
Chevy
Chevy
Ford
Audi
Tesla
Ford
Chevy



Make ID
0
0
1
2
3
1
0

Dictionary

ID	Make
0	Chevy
1	Ford
2	Audi
3	Tesla

Huffman Coding

- An optimal prefix algorithm
- More-common values are represented using fewer bits

Dictionary

ID	Make	Bits Req'd
0	Chevy	1
1	Ford	1
2	Audi	2
3	Tesla	2
…		
95	Bugatti	7

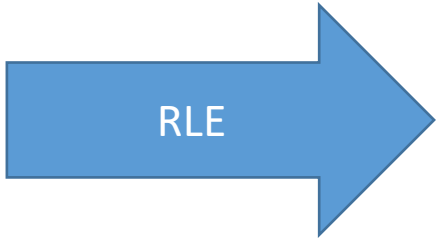
Run Length Encoding (RLE)

- Avoids repetition by grouping same values together

Color
Red
Red
…
Blue
Blue
Blue
…

105x

92x

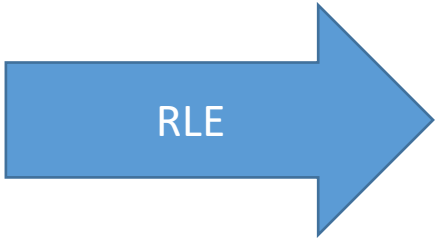


Color	Count
Red	105
Blue	92

Run Length Encoding (RLE)

- Can be less efficient depending on ordering

Color
Red
Red
Blue
Red
Blue
Blue
Blue



Color	Count
Red	2
Blue	1
Red	1
Blue	3

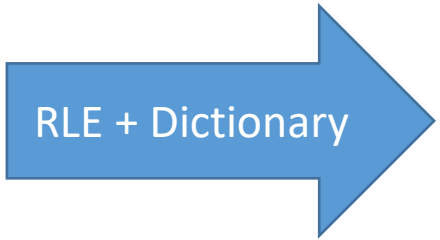
Run Length Encoding (RLE)

- Can combine with dictionary

Color
Red
Red
Blue
Red
Blue
Blue
Blue

Dictionary

ID	Color
0	Red
1	Blue



Color ID	Count
0	2
1	1
0	1
1	3

Columnstore Archival Compression

- The one option you do have for columnstore compression
  - Further compresses the columnstore index to save space
  - Requires more time for storage & retrieval
- Adds another layer of compression
  - Modified LZ77?

When Should Columnstore Be Used?

- Queries that scan large amounts of data
  - Especially on large tables
- Datasets with highly-repetitive values
- Data Warehousing Workloads
  - Fact Tables
  - Large Dimension Tables
- Real-time analysis on OLTP workloads

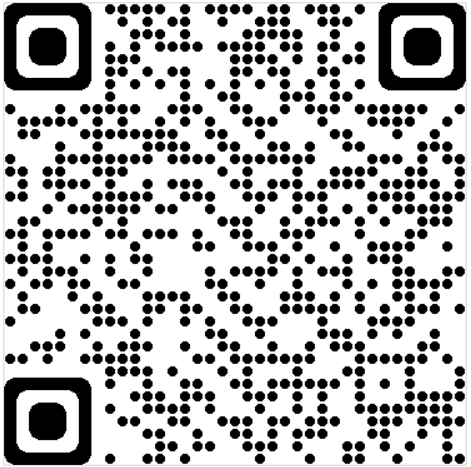
When Should Columnstore Be Avoided?

- “Small tables” (under 1M rows)
  - Won’t benefit from compression or be stored as segments
- Columns containing highly-unique strings
- Queries that usually return all rows in a table
  - Data won’t be returned any faster by a columnstore index
- Frequently updated tables

Resources

- Jonathan Kehayias shows how to use XE to track compression operations: http://www.sqlskills.com/blogs/jonathan/post/An-XEvent-a-Day-(28-of-31)-e28093-Tracking-Page-Compression-Operations.aspx
- Sanjay Mishra whitepaper on Data Compression strategy & best practices: http://msdn.microsoft.com/en-us/library/dd894051(SQL.100).aspx
- MCM Readiness video on SQL Server data compression: http://technet.microsoft.com/en-us/sqlserver/gg313758.aspx
- Niko Neugebauer on columnstore indexes: http://www.nikoport.com/columnstore/
- Materials from this session, more resources, etc: https://www.bobpusateri.com/r/Compression

Questions? THANK YOU FOR ATTENDING!



Evaluations!

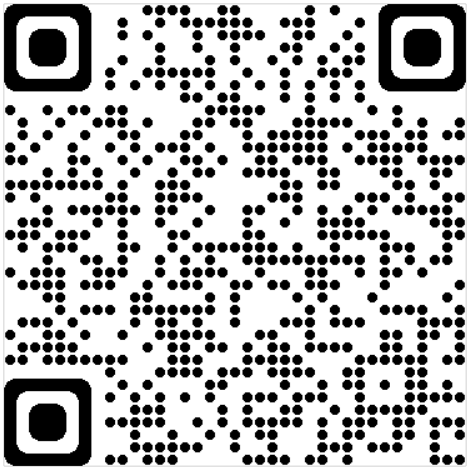
evals.datagrillen.com

@sqlbob

bob@bobpusateri.com

bobpusateri.com

bobpusateri



Session Materials!

www.bobpusateri.com/r/Compression