Data Blocks: Hybrid OLTP and OLAP on compressed storage

Ben Brümmer
Technische Universität München
Fürstenfeldbruck, 26. November 2018
Problem

• HDD/Archive/Tape-Storage and even SSDs are slow
• Main memory is small
• Hybrid OLTP and OLAP databases should have high throughput
Solution

Design a storage format for cold data with:
• Compression (fit into main memory)
• High point accessibility, mainly for OLTP but also OLAP requests (fast access)

=> Data Blocks
How are Data Blocks constructed?

1. # of tuples
2. Information about each attribute
   • Compression method
   • Offsets
3. SMA and PSMA for 1. attribute
4. The actual data for attribute 1
5. Etc

[1] Paper: Figure 3
Compression

Different method chosen based on value distribution (per attribute)

Three different compression methods:
• Single value compression
• Ordered dictionary compression
• Truncation

➔ Possibility of point access (no sub-byte encoding)
➔ SIMD operations are usable on compressed data
Compression

<table>
<thead>
<tr>
<th>Data Base</th>
<th>data</th>
<th>Uncompressed</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vectorwise</td>
<td>TPC-H 100</td>
<td>105 GB</td>
<td>66 GB</td>
</tr>
<tr>
<td></td>
<td>IMDB Cast</td>
<td>0.72 GB</td>
<td>0.24 GB</td>
</tr>
<tr>
<td></td>
<td>Flights</td>
<td>11 GB</td>
<td>3.2 GB</td>
</tr>
<tr>
<td>Hyper</td>
<td>TPC-H 100</td>
<td>126 GB</td>
<td>66 GB</td>
</tr>
<tr>
<td></td>
<td>IMDB Cast</td>
<td>1.8 GB</td>
<td>0.5 GB</td>
</tr>
<tr>
<td></td>
<td>Flights</td>
<td>21 GB</td>
<td>4.2 GB</td>
</tr>
</tbody>
</table>
Compression

[1] Paper: Figure 10
SMA and PSMA

- SMA means Small Materialized Aggregate
  - Minimum and maximum for each attribute
  - Used for evaluation if a block can be skipped

- PSMA is a positional SMA
  - Maps a value to a range in the data block
  - Uses a lookup table
PSMA

- Light weight indexing structure
- Improve scan ranges by checking lookup table
- Table computed when creating a data block

- For each byte of attribute data type $2^8$ entries, e.g. 4-byte int $\rightarrow 4 \times 2^8$ entries
- Each entry is a range
Building a lookup table

- Initialize table with 0 ranges

SMA min: 2

Delta bytes

Attribute data

Lookup table

# of values mapping to entry
Building a lookup table

- Probe value from attribute e.g. 5 at i = 0
- Only left most non 0 byte important

SMA min: 2

Delta bytes

Attribute data

# of values mapping to entry

0  [0, 0)  1
1  [0, 0)  1
2  [0, 0)  1
3  [0, 0)  1
...
257 [0, 0)  $2^{28}$
...
1024 [0, 0)  $2^{24}$
Building a lookup table

- Evaluate delta
- If range at delta is 0-range: update to \([i, i+1)\)

SMA min: 2   \(5 - \text{min} = 3\)

Delta bytes

\[
\begin{array}{cccc}
00 & 00 & 00 & 03 \\
\end{array}
\]

Attribute data

\[
\begin{array}{ccccccc}
i=0 & 5 & 20 & 2 & 4 & 290 & 8 & 5 & 420 & 3 & 10 \\
\end{array}
\]

# of values mapping to entry

\[
\begin{array}{cccc}
0 & [0, 0) & 1 \\
1 & [0, 0) & 1 \\
2 & [0, 0) & 1 \\
3 & [0, 1) & 1 \\
\ldots \\
257 & [0, 0) & 2^{8} \\
\ldots \\
1024 & [0, 0) & 2^{24} \\
\end{array}
\]

Lookup table
Building a lookup table

- Evaluate delta
- If range at delta is 0-range: update to \([ i, i+1)\)

SMA min: 2  
290 – min = 288

**Delta bytes**

0x01 + 256*1

**Attribute data**

5  20  2  4  290  8  5  420  3  10

**Lookup table**

<table>
<thead>
<tr>
<th>Entry</th>
<th>Range</th>
<th># of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[2, 3]</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>[0, 0]</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[3, 4]</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>[0, 1]</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>257</td>
<td>[4, 5]</td>
<td>(2^8)</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>[0, 0]</td>
<td>(2^{24})</td>
</tr>
</tbody>
</table>
Building a lookup table

- Evaluate delta
- If range at delta not 0-range: update later to $i+1$

SMA min: 2

$5 - \text{min} = 3$

Delta bytes

| 00 | 00 | 00 | 03 |

Attribute data

| 5  | 20 | 2  | 4  | 290 | 8  | 5  | 420 | 3  | 10 |

# of values mapping to entry

| 0  | [2, 3) | 1  |
| 1  | [0, 0) | 1  |
| 2  | [3, 4) | 1  |
| 3  | [0, 7) | 1  |
| ... |         | 1  |
| 257 | [4, 5) | $2^8$ |
| ... |         |     |
| 1023 | [0, 0) | $2^{24}$ |

Lookup table
Building a lookup table

- Evaluate delta
- If range at delta not 0-range: update later to i+1

SMA min: 2

420 – min = 418

Delta bytes

0x01 + 256*1

Attribute data

| 5 | 20 | 2 | 4 | 290 | 8 | 5 | 420 | 3 | 10 |

Lookup table

| 0  | [ 2 , 3 ) | 1 |
| 1  | [ 0 , 0 ) | 1 |
| 2  | [ 3 , 4 ) | 1 |
| 3  | [ 0 , 7 ) | 1 |
| ... | ... | ... |
| 257 | [ 4 , 8 ) | $2^8$ |
| ... | ... | ... |
| 1024 | [ 0 , 0 ) | $2^{24}$ |
Building a lookup table

- finished lookup table when all data was probed

SMA min: 2

<table>
<thead>
<tr>
<th>Delta bytes</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Attribute data</th>
<th>5</th>
<th>20</th>
<th>2</th>
<th>4</th>
<th>290</th>
<th>8</th>
<th>5</th>
<th>420</th>
<th>3</th>
<th>10</th>
</tr>
</thead>
</table>

# of values mapping to entry

<table>
<thead>
<tr>
<th>#</th>
<th>[low, high]</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[2, 3)</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>[8, 9)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[3, 4)</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>[0, 7)</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>257</td>
<td>[4, 8)</td>
<td>2^8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1023</td>
<td>[0, 0)</td>
<td>2^24</td>
</tr>
</tbody>
</table>

Lookup table
Using the lookup table

- when SMA does not rule out block
- calculate delta, check table, return scan range

SMA min: 2
e.g. 344:
344 – min = 342

<table>
<thead>
<tr>
<th>Delta bytes</th>
<th>Lookup table</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 00 01 56</td>
<td>0  [2, 3) 1</td>
</tr>
<tr>
<td></td>
<td>1  [8, 9) 1</td>
</tr>
<tr>
<td></td>
<td>2  [3, 4) 1</td>
</tr>
<tr>
<td></td>
<td>3  [0, 7) 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>257 [4, 8) 2^8</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>1023 [0, 0) 2^34</td>
</tr>
</tbody>
</table>

Attribute data

0x01 + 256*1

Range: [4,8]
PSMA

- used to further narrow scan ranges
- lookup on multiple attributes: intersecting of returned ranges
- very useful on sorted data sets
- works better for small delta
- building table is $O(n)$ per attribute
Lookup on a Data Block

1) Check SMA
2) Check PSMA
3) Depending on compression method and predicate block may still be ruled out
   - e.g. ordered dictionary and “equal” - > binary search on dictionary
4) Evaluate restrictions on compressed Data - > Return vector with offset to tuples
5) Push tuples into consuming operator (e.g. join)
   - vector at a time
6) Repeat until no matches in Data Block

Small overhead for converting restriction constants into compressed version
Further improvements

- Just in Time query optimizations
- Same interface for scans on hot and cold (compressed) data
- SIMD accelerated match finding
## Benchmarks

<table>
<thead>
<tr>
<th>scan type</th>
<th>geometric mean</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>JIT uncompressed</td>
<td>0.586s</td>
<td>21.7s</td>
</tr>
<tr>
<td>Data Blocks</td>
<td>0.555s (1.06x)</td>
<td>21.5s</td>
</tr>
<tr>
<td>+PSMA</td>
<td>0.463s (1.27x)</td>
<td>20.2s</td>
</tr>
<tr>
<td>Vectorwise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uncompressed</td>
<td>2.336s</td>
<td>74.4s</td>
</tr>
<tr>
<td>compressed</td>
<td>2.527s (0.92x)</td>
<td>78.5s</td>
</tr>
</tbody>
</table>

Runtimes of TPC-H Queries on Hyper and Vectorwise in different storage formats
Benchmarks

Figure 11: Speedup of TPC-H Q6 (scale factor 100) on block-wise sorted data (+SORT)

[1] Paper: Figure 11
Benchmarks – Shuffled Data

- Uncompressed
- Data Blocks
- +PSMA

- PK index
- no Index

lookups/s
Benchmarks – Ordered Data

- uncompressed
- Data Blocks
- +PSMA
Sources

(1) Data Blocks: Hybrid OLTP and OLAP on Compressed Storage using both Vectorization and Compilation
(2) Slides: Data Blocks by Harald Lang