Group-Key Indices
Lookup and Maintenance

Martin Faust, David Schwalb, Jens Krueger and Hasso Plattner

Hasso Plattner Institute, Potsdam, Germany
at ADMS 2012
Agenda

- Background
  - Motivation
- Dictionary encoded In-Memory Column Store
- Group-Key Indices
  - Structure & Lookup
  - Creation from Scratch
  - Integration of Maintenance into the Merge Process
- Performance Results
Motivation

- Indices to speed up query processing of selections
- Memory traffic to assess lookup and maintenance costs
- Integration with merge process to minimize maintenance costs

Query distribution by elapsed time (Customer System)

- Selects: 87.6%
- Inserts: 11.3%
- Updates: 0.9%
- Deletes: 0.2%
In-Memory Column Store

- Column consists of typically large, read-only main partition and smaller, writable delta partition
- Insert-only mode
- Queries are executed on main and delta partition simultaneously
Main Partition

- The value domain is encoded in a **sorted dictionary**
- Column is stored as vector of bit-packed value-ids, called the **attribute vector**
- High compression and high scan speed
Delta Partition

- Delta partition’s values are stored **uncompressed** in a vector \((D)\)
- **CSB+ tree index** stores value to position information \((T)\)
- Fast operations on small delta partition
Merge Process

• Delta is uncompressed and therefore large
• The performance decreases, if the delta partition grows too large
• Merge main and delta partition from time to time to create a new main partition
Group-Key Index Structure I

- Inverted index
- Index is only applied to the main partition
- Index maps value-ids to positions
- Index consists of two bit-packed structures: index offsets (I) and index postings (P)
Group-Key Index Structure II

\[ E_C^j = \lceil \log_2 |U_M^j| \rceil \]
We will investigate the implications on the viability of index usage in Section 4.3.

The Group-Key index consists of two separate structures: the sorted main partition and the bit-packed attribute vector and writing the positions. Because no inherent quantification of the performance impact.

\[
E_C^j = \lceil \log_2 |U_M^j| \rceil
\]

\[
A^j = \lceil \log_2 (N_M) \rceil \text{ bits}
\]

\[
\text{sizeof}(I^j) = (|U_M^j| + 1) \cdot \frac{A^j}{8} \text{ bytes}
\]

\[
\text{sizeof}(P^j) = N_M \cdot \frac{A^j}{8} \text{ bytes}
\]
Group-Key Index Structure II

<table>
<thead>
<tr>
<th>$V^j_M$</th>
<th>$U^j_M$</th>
<th>$I^j$</th>
<th>$P^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Attribute Vector</strong></td>
<td><strong>Main Dictionary</strong></td>
<td><strong>Index Offsets</strong></td>
<td><strong>Index Postings</strong></td>
</tr>
<tr>
<td>0: hotel</td>
<td>0: apple</td>
<td>0:</td>
<td>4: apple</td>
</tr>
<tr>
<td>1: delta</td>
<td>1: charlie</td>
<td>1: 2:</td>
<td>5: charlie</td>
</tr>
<tr>
<td>2: frank</td>
<td>2: delta</td>
<td>2: 3:</td>
<td>6: charlie</td>
</tr>
<tr>
<td>3: delta</td>
<td>3: frank</td>
<td>3: 5:</td>
<td>7: charlie</td>
</tr>
<tr>
<td>4: apple</td>
<td>4: hotel</td>
<td>4: 6:</td>
<td></td>
</tr>
<tr>
<td>5: charlie</td>
<td>5: inbox</td>
<td>5: 7:</td>
<td></td>
</tr>
<tr>
<td>6: charlie</td>
<td>6: hotel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7: inbox</td>
<td>7: inbox</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$E_C^j = \lceil \log_2 |U^j_M| \rceil$$

$$A^j = \lceil \log_2 (N_M) \rceil \text{ bits}$$

$$\text{sizeof}(I^j) = (|U^j_M| + 1) \cdot \frac{A^j}{8} \text{ bytes}$$

$$\text{sizeof}(P^j) = N_M \cdot \frac{A^j}{8} \text{ bytes}$$

Martin Faust | ADMS 2012
We now define how the structures of the Group-Key Index are compressed. At least 1 bit (only two distinct values), the posting structure’s maximum size in relation to the row count of the table. We define the size of an attribute vector in a bit-packed format as $E_C^j = \lceil \log_2 |U_M^j| \rceil$.

$$A^j = \lceil \log_2 (N_M) \rceil \text{ bits}$$

$$\text{sizeof}(I^j) = (|U_M^j| + 1) \cdot \frac{A^j}{8} \text{ bytes}$$

$$\text{sizeof}(P^j) = N_M \cdot \frac{A^j}{8} \text{ bytes}$$

In particular, the interval in Equation 1 contains all positions that the resulting list is ordered for a single-value predicate. For a value-id $v$, in Figure 1. As an example, the steps that are necessary to form a position list is necessary.

In this Section we present a dense, bit-packed Group-Key index-aware column merge process and the proposed rebuilding algorithm.
Group-Key Index Structure II

<table>
<thead>
<tr>
<th>$V_M^j$</th>
<th>$U_M^j$</th>
<th>$P^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Attribute Vector</strong></td>
<td><strong>Main Dictionary</strong></td>
<td><strong>Index Postings</strong></td>
</tr>
<tr>
<td>hotel</td>
<td>apple</td>
<td>4</td>
</tr>
<tr>
<td>delta</td>
<td>delta</td>
<td>5</td>
</tr>
<tr>
<td>frank</td>
<td>frank</td>
<td>3</td>
</tr>
<tr>
<td>delta</td>
<td>charlie</td>
<td>6</td>
</tr>
<tr>
<td>apple</td>
<td>hotel</td>
<td>0</td>
</tr>
<tr>
<td>charlie</td>
<td>inbox</td>
<td></td>
</tr>
</tbody>
</table>

$E_C^j = \lceil \log_2 |U_M^j| \rceil$

$A^j = \lceil \log_2 (N_M) \rceil \text{ bits}$

$\text{sizeof}(I^j) = (|U_M^j| + 1) \cdot \frac{A^j}{8} \text{ bytes}$

$\text{sizeof}(P^j) = N_M \cdot \frac{A^j}{8} \text{ bytes}$
We now define how the structures of the Group-Key Index are compressed. Like the Start-Stop Index briefly in Section 3.3.2.

Accordingly, after the delta and main partition are merged the indexing process consists of three partitions we run an indexing process after the execution of a column without an index, or for a newly merged main partition, as shown in Equation 2.

The Group-Key index consists of two separate structures: the Group-Key index. Step 1 and Step 3 are dependent on the length of the dictionary, each executes in \( O \left( \frac{M}{8} \right) \) bytes (3).

The relationship between the involved structures is shown in Figure 3. Both structures of the Group-Key index are bit-packed, then the according o

\[ A^j = \left\lfloor \log_2 (N_M) \right\rfloor \text{ bits} \]

\[ \text{sizeof}(I^j) = (|U_M^j| + 1) \cdot \frac{A^j}{8} \text{ bytes} \]

\[ \text{sizeof}(P^j) = N_M \cdot \frac{A^j}{8} \text{ bytes} \]

\[ E_C^j = \left\lfloor \log_2 |U_M^j| \right\rfloor \]
### 3.1.2 Compression of the Group-Key Index

We will investigate the implications on the viability of index usage in Section 4.3.

---

<table>
<thead>
<tr>
<th>(V^j_M)</th>
<th>(U^j_M)</th>
<th>(P^j)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Attribute Vector</strong></td>
<td><strong>Main Dictionary</strong></td>
<td><strong>Index Postings</strong></td>
</tr>
<tr>
<td>0: hotel</td>
<td>0: apple</td>
<td>4: apple</td>
</tr>
<tr>
<td>1: delta</td>
<td>1: charlie</td>
<td>5: charlie</td>
</tr>
<tr>
<td>2: frank</td>
<td>2: delta</td>
<td>6: charlie</td>
</tr>
<tr>
<td>3: delta</td>
<td>3: frank</td>
<td>7: charlie</td>
</tr>
<tr>
<td>4: apple</td>
<td>4: hotel</td>
<td></td>
</tr>
<tr>
<td>5: charlie</td>
<td>5: inbox</td>
<td></td>
</tr>
<tr>
<td>6: charlie</td>
<td>6: hotel</td>
<td></td>
</tr>
<tr>
<td>7: inbox</td>
<td>7: inbox</td>
<td></td>
</tr>
</tbody>
</table>

\[ E^j_C = \lceil \log_2 |U^j_M| \rceil \]

\[ A^j = \lceil \log_2 (N_M) \rceil \text{ bits} \]

\[ \text{sizeof}(I^j) = (|U^j_M| + 1) \cdot \frac{A^j}{8} \text{ bytes} \]

\[ \text{sizeof}(P^j) = N_M \cdot \frac{A^j}{8} \text{ bytes} \]
Lookup Performance

Index Lookup vs. Column Scan ($N_M = 30000000$)

Y1-Axis
- Column Scan
- Index Lookup

Y2-Axis
- Index Scan
- Column Scan

Distinct Value Fraction

CPU Cycles

Monday, August 27, 12
### Merge Process

**Main Partition**
- \( \mathbf{M}^j \)
- Data: \( V^j_M \)
- Index: \( U^j_M \)
- \( P^j \)

**Delta Partition**
- \( \mathbf{D}^j \)
- \( T^j \)

**New Main Partition**
- \( \mathbf{M}'^j \)
- Data: \( V'^j_{M'} \)
- Index: \( U'^j_{M'} \)
- \( P'^j \)

\( \mathbf{M}^j \) and \( \mathbf{M}'^j \) are connected through the merge process, which incorporates changes from \( \mathbf{D}^j \) into \( \mathbf{M}^j \) to form \( \mathbf{M}'^j \).
Group-Key Index
Creation from Scratch

Column Merge
1. Transform delta partition, merge dictionaries
2. Update main partition’s value-ids

Index Creation
1. Counting occurrences of value-ids
2. Creating offsets for P in I
3. Creating postings in P

Table 2.1.

<table>
<thead>
<tr>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of Address in Main Partition</td>
<td>bits</td>
</tr>
<tr>
<td>Updated main attribute vector</td>
<td></td>
</tr>
<tr>
<td>Bucket Pointer List / Buckets</td>
<td></td>
</tr>
<tr>
<td>Cache Line size</td>
<td>bytes</td>
</tr>
<tr>
<td>Compressed Value-Length</td>
<td>bits</td>
</tr>
<tr>
<td>Uncompressed Value-Length</td>
<td>bytes</td>
</tr>
<tr>
<td>Sorted dictionary of the main/delta partition</td>
<td></td>
</tr>
<tr>
<td>Main/delta attribute vector of the sets / Postings</td>
<td></td>
</tr>
<tr>
<td>Number of tuples in the main/delta partition</td>
<td></td>
</tr>
<tr>
<td>Number of tuples in the updated table</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.11.
Integrate with Merge Process

- Merge index together with dictionaries
- Old Group-Key Index and CSB+ Index on Delta Partition are combined

```
<table>
<thead>
<tr>
<th>P^j</th>
<th>I^j</th>
<th>P^j</th>
<th>I^j</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_M^j</td>
<td>X_M^j</td>
<td>U_M^j</td>
<td>X_M^j</td>
</tr>
<tr>
<td>V_D</td>
<td>V_M^j</td>
<td>U_M</td>
<td>X_M</td>
</tr>
<tr>
<td>T^j</td>
<td>V_M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Group-Key Index-aware Merge Process Strategy 3

- Sequential Read
- Sequential Write
- Random Read
- Random Write

Martin Faust | ADMS 2012
Chapter 2. Index Types

The logical itemcount of Index O

Number of tuples in the updated table - with one additional value to mark the end. In the current implementation the dated) entry.

Table 2.1.

<table>
<thead>
<tr>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket Pointer List / Buckets</td>
<td></td>
</tr>
<tr>
<td>Delta Frequencies</td>
<td></td>
</tr>
<tr>
<td>Sorted dictionary of the main/delta partition</td>
<td></td>
</tr>
<tr>
<td>Updated main attribute vector</td>
<td></td>
</tr>
<tr>
<td>Main/delta attribute vector of the merged column</td>
<td></td>
</tr>
<tr>
<td>Merged column</td>
<td></td>
</tr>
<tr>
<td>Uncompressed Value-Length bytes</td>
<td></td>
</tr>
<tr>
<td>Compressed Value-Length bits</td>
<td></td>
</tr>
<tr>
<td>Compressed Value-Length after merge bits</td>
<td></td>
</tr>
</tbody>
</table>

Symbol Definition. Entities annotated with e

...sets / Postings -

Symbol Definition. Entities annotated with e

is determined by the size of the dictionary

for the first or last value have to be considered. The values in...

Value-id

Martin Faust | ADMS 2012
Chapter 2. Index Types

Sorted dictionary of the main/delta partition -
Main/delta partition of the
Number of tuples in the main/delta partition -

Symbol Definition. Entities annotated with
Compressed Value-Length after merge bits

Memory Traces

Compressed Value-Length bits

Extended auxiliary structure for delta -
Auxiliary structure for the main/delta -
Compressed Value-Length after merge bits

Δ Frequencies -
Extended auxiliary structure for delta -

Number of tuples in the updated table -

Main/delta attribute vector of the
Description Unit

is determined by the size of the dictionary
for the first or last value have to be considered. The values in
...
Chapter 2. Index Types

The logical itemcount of Cache Line size bytes

Bucket Pointer List / Buckets -

Index O

Extended auxiliary structure for delta -

Length of Address in Main Partition bits

For a given column

first value is always 0, which allows for easier query code, since no edge cases

dated) entry.

Table 2.1.
### Table 2.1

<table>
<thead>
<tr>
<th>Dictionary Entries</th>
<th>Postings offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>0</td>
</tr>
<tr>
<td>charlie</td>
<td>1</td>
</tr>
<tr>
<td>delta</td>
<td>3</td>
</tr>
<tr>
<td>frank</td>
<td>5</td>
</tr>
<tr>
<td>hotel</td>
<td>6</td>
</tr>
<tr>
<td>inbox</td>
<td>7</td>
</tr>
</tbody>
</table>

**Uncompressed Value-Length after merge bits**: 

- Main: \[ \text{Value ID} \]
  - Value ID: 0, 1, 2, 3, 4, 5, 6, 7.
  - Size: 8

**Updated Index Object**

- \( I^j \)
  - Postings offset: 0, 1, 2, 3.

- \( U^j \)
  - Dictionary Entries: apple, charlie, delta, frank, hotel, inbox.
  - Size: 8

**Delta Frequencies**

- Fraction of unique values in main/delta:
  - Updated main attribute vector:
    - Merged column:
      - Updated main dictionary:
        - Main/delta partition of the bucket pointer list/buckets:
          - Compressed Value-Length bits:
            - Uncompressed Value-Length bytes:
              - Length of Address in Main Partition bits:
                - Dictionary Entries (corresponding tuples) = \[ 1, 3, 5, 7, 9, 11, 12 \]

**Example**

- For a given column:
  - First value is always 0, which allows for easier query code, since no edge cases.

**Outputs**

- \( U'_{M}^{j} \)
  - Postings offset: 0
  - Read offset: 0

- \( I'^{j} \)
  - Position in Attribute Vector:
    - Position: 4
    - Value: \[ 0, 1, 2, 3, 4, 5, 6, 7 \]

**X^j_M**

- New Value-id: 0

---

**Tuples**

- Number of tuples in the main/delta partition: 8

---

**Dictionary Entries**

- \( V^i_M \)
  - Value ID: 0, 1, 2, 3, 4, 5, 6, 7.
  - Size: 8

**Delta Position Lists**

- Uncompressed:
  - bodo
  - hotel
  - frank
- CSB + Leaves:
  - bodo
  - hotel
  - frank

---

**Martin Faust | ADMS 2012**

Monday, August 27, 12
**Inputs**

- **\(V^j_M\)**: Value ID
  - Value: 4, 2, 3, 2, 0, 1, 1, 5
  - Size: 8

- **\(U^j_M\)**: Dictionary Entries
  - Entries: apple, charlie, delta, frank, hotel, inbox

- **\(I^j\)**: Postings offset
  - Offset: 0, 1, 3, 5, 6, 7, 8

**Outputs**

- **\(\Upsilon^j_M\)**: Postings
  - Entries: apple, hotel, delta

- **\(\Gamma^j\)**: Read offset
  - Offset: 0, 1, 3, 5

- **\(X^j_M\)**: New Value-id
  - Value: 0

- **\(D^j\)**: CSB+ Leaves
  - Entries: bodo, hotel, frank

- **\(\Delta^j\)**: Delta Position Lists
  - Lists: bodo, hotel, frank

**C (corresponding tuples) = \(0 \; 1 \; 3 \; 5 \; 7 \; 9 \; 11 \; 12\)**

Martin Faust | ADMS 2012

Monday, August 27, 12
Table 2.1.

<table>
<thead>
<tr>
<th>Dictionary Entries</th>
<th>Postings Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>0</td>
</tr>
<tr>
<td>charlie</td>
<td>1</td>
</tr>
<tr>
<td>delta</td>
<td>2</td>
</tr>
<tr>
<td>frank</td>
<td>3</td>
</tr>
<tr>
<td>hotel</td>
<td>4</td>
</tr>
<tr>
<td>inbox</td>
<td>5</td>
</tr>
</tbody>
</table>

Delta Position Lists

<table>
<thead>
<tr>
<th>Uncompressed</th>
<th>Delta Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodo</td>
<td>0, 1</td>
</tr>
<tr>
<td>hotel</td>
<td>2, 3</td>
</tr>
<tr>
<td>frank</td>
<td>4</td>
</tr>
</tbody>
</table>

Auxiliary structure for the main/delta

\[ \frac{1}{12} \]
increasing, all positive, and less or equal than

12 Chapter 2. Index Types

Bucket Pointer List / Buckets -
Extended auxiliary structure for delta -
Fraction of unique values in main/delta -
Updated main dictionary -
Sorted dictionary of the main/delta partition -
Merged column -
For a given column

Table 2.1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V^j_M$</td>
<td>Value ID</td>
<td></td>
</tr>
<tr>
<td>$U^j_M$</td>
<td>Dictionary Entries</td>
<td></td>
</tr>
<tr>
<td>$I^j$</td>
<td>Postings offset</td>
<td></td>
</tr>
<tr>
<td>$U'^j_M$</td>
<td>Postings</td>
<td></td>
</tr>
<tr>
<td>$I'^j$</td>
<td>Read offset</td>
<td></td>
</tr>
<tr>
<td>$X^j_M$</td>
<td>New Value-id</td>
<td></td>
</tr>
</tbody>
</table>

Cache Line size bytes
Uncompressed Value-Length bytes
Compressed Value-Length bits
Number of tuples in the main/delta partition -
Number of columns in the table -

...
increasing, all positive, and less or equal than the first value is always 0, which allows for easier query code, since no edge cases need to be considered. The values in the sorted dictionary of the main/delta partition are used to generate the delta frequencies.

Memory Trajectory: The memory trajectory includes the cache line size, index offset, and length of address in the main partition. The updated main dictionary and the merged column are also part of the memory trajectory.

Delta Frequencies: The delta frequencies are calculated for each value in the dictionary entries. The values are compressed to reduce storage space.

Extended Auxiliary Structure: An extended auxiliary structure is used for delta operations, which includes the updated main attribute vector and the merged column.

Symbol Definition: Entities annotated with specific symbols are used to represent the merged (updated) entry.
### Outputs

**V_M^j**
- Value ID

<table>
<thead>
<tr>
<th>Value ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

**U_M^j**
- Dictionary Entries

<table>
<thead>
<tr>
<th>Dictionary Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
</tr>
<tr>
<td>charlie</td>
</tr>
<tr>
<td>delta</td>
</tr>
<tr>
<td>frank</td>
</tr>
<tr>
<td>hotel</td>
</tr>
<tr>
<td>inbox</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dictionary Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
</tr>
<tr>
<td>charlie</td>
</tr>
</tbody>
</table>

**I^j**
- Postings offset

<table>
<thead>
<tr>
<th>Postings offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

**X_M^j**
- New Value-id

<table>
<thead>
<tr>
<th>New Value-id</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

**D^j**
- CSB + Leaf Lists

<table>
<thead>
<tr>
<th>Uncompressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodo</td>
</tr>
<tr>
<td>hotel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uncompressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Delta Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

**P^j**
- Position in Attribute Vector

<table>
<thead>
<tr>
<th>Position in Attribute Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 8 9 5 6</td>
</tr>
</tbody>
</table>

**P'j**
- Postings Read offset

<table>
<thead>
<tr>
<th>Postings Read offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

**M**
- (corresponding tuples)

<table>
<thead>
<tr>
<th>(corresponding tuples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 3 5 7 9 11 12</td>
</tr>
</tbody>
</table>

**C**
- Number of tuples in the main/delta partition

<table>
<thead>
<tr>
<th>Number of tuples in the main/delta partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
</tr>
</tbody>
</table>

**Delta Frequencies**
- Length of Address in Main Partition bits

<table>
<thead>
<tr>
<th>Length of Address in Main Partition bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**Main/delta attribute vector**

<table>
<thead>
<tr>
<th>Main/delta attribute vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**Main/delta partition**

<table>
<thead>
<tr>
<th>Main/delta partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**Memory Tra**

<table>
<thead>
<tr>
<th>Memory Tra</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
</tr>
</tbody>
</table>

**Sorted dictionary of the main/delta partition**

<table>
<thead>
<tr>
<th>Sorted dictionary of the main/delta partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6</td>
</tr>
</tbody>
</table>

**Symbol Definition**
- Entities annotated with `s`
- `j`
- `P`
increasing, all positive, and less or equal than 12

Chapter 2. Index Types

Cache Line size bytes

For the first or last value have to be considered. The values in

with one additional value to mark the end. In the current implementation the

Table 2.1.

Fraction of unique values in main/delta -

For a given column

Number of columns in the table -

The logical itemcount of

Memory Tra

Cache Line size bytes

Compressed Value-Length bits

Extended auxiliary structure for delta -

Compressed Value-Length after merge bits

Uncompressed Value-Length bytes

Updated main dictionary -

Sorted dictionary of the main/delta partition -

Updated main attribute vector -

Number of tuples in the updated table -

Number of tuples in the main/delta partition -

Symbol Definition. Entities annotated with

\[ P_j \]

\[ I_j \]

\[ D_j \]

\[ U_j \]

\[ X_j \]

\[ C \]

(comparing tuples) = 0 1 3 5 7 9 11 12

Martin Faust | ADMS 2012

Monday, August 27, 12
```plaintext
<table>
<thead>
<tr>
<th>$V_M^j$</th>
<th>$U_M^j$</th>
<th>$I^j$</th>
<th>$U'_M^j$</th>
<th>$I'_j$</th>
<th>$X_M^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value ID</td>
<td>Dictionary Entries</td>
<td>Postings offset</td>
<td>Postings offset</td>
<td>Read offset</td>
<td>New Value-id</td>
</tr>
<tr>
<td>0</td>
<td>apple</td>
<td>0</td>
<td>apple</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>charlie</td>
<td>1</td>
<td>bodo</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>delta</td>
<td>3</td>
<td>charlie</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>frank</td>
<td>5</td>
<td>delta</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>hotel</td>
<td>6</td>
<td>frank</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>inbox</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Size = 8
```

---

**Diagram:**
- **Outputs** column:
  - **$U'_M^j$** for postings.
  - **$I'_j$** for read offset.
  - **$X_M^j$** for new value-id.

- **$D^j$ column:**
  - **Uncompressed** row:
    - bodo, hotel
  - **Compressed** row:
    - bodo, hotel

- **$V_M^j$ column:**
  - Value ID:
    - 0, 1, 2, 3, 4, 5, 6, 7

- **$I^j$ column:**
  - Postings offset:
    - 0, 1, 3, 5, 6, 7

- **$U_M^j$ column:**
  - Dictionary Entries:
    - apple, charlie, delta, frank, hotel, inbox

- **$P_j$ column:**
  - Position in Attribute Vector:
    - 4, 5, 6, 1, 3, 2

- **$C$ column:**
  - (corresponding tuples):
    - 0, 1, 3, 5, 7, 10, 11, 12

---

**Martin Faust | ADMS 2012**

Monday, August 27, 12
increasing, all positive, and less or equal than

Chapter 2. Index Types

Merged column -

Description Unit

first value is always 0, which allows for easier query code, since no edge cases

Table 2.1.

Cache Line size bytes

Index O

Delta Frequencies -

Fraction of unique values in main/delta -

Compressed Value-Length after merge bits

Uncompressed Value-Length bytes

Merged column -

For a given column

sets / Postings -

is determined by the size of the dictionary

Symbol Definition. Entities annotated with

...}

Symbol

E

...}

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E

E
For a given column the first or last value have to be considered. The values in the `dated` entry.

### Table 2.1.

<table>
<thead>
<tr>
<th>Column</th>
<th>Value ID</th>
<th>Dictionary Entries</th>
<th>Postings Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>apple</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>charlie</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>delta</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>frank</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>hotel</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>inbox</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Updated Index

- **Merged column**
- **Main/delta partition of the main attribute vector**
- **Auxiliary structure for the main/delta**
- **Extended auxiliary structure for delta**
- **Updated main dictionary**
- **Compressed Value-Length after merge bits**
- **Uncompressed Value-Length bytes**
- **Fraction of unique values in main/delta**
- **Logical itemcount of main/delta**
- **Number of tuples in the main/delta partition**

### Symbol Definition

- Entities annotated with `symbol`:
  - `sets` / `Postings`
  - `Bucket Pointer List / Buckets`
  - `Main/delta attribute vector`
  - `Main/delta partition of the main attribute vector`
  - `Auxiliary structure for the main/delta`
  - `Extended auxiliary structure for delta`
  - `Updated main dictionary`
  - `Compressed Value-Length bits`
  - `Uncompressed Value-Length bytes`
  - `Fraction of unique values in main/delta`
  - `Logical itemcount of main/delta`
  - `Number of tuples in the main/delta partition`
increasing, all positive, and less or equal than 12 Chapter 2. Index Types

Delta Frequencies -
Length of Address in Main Partition bits
Compressed Value-Length bits
Updated main attribute vector -
Description Unit

first value is always 0, which allows for easier query code, since no edge cases
Sorted dictionary of the main/delta partition -

Memory Trace
Cache Line size bytes
Index O

Auxiliary structure for the main/delta -
Fraction of unique values in main/delta -

Merged column -

Extended auxiliary structure for delta -
Length of Address in Main Partition bits
Uncompressed Value-Length bytes
Updated main dictionary -

Table 2.1.

Symbol Definition. Entities annotated with

...
### Table 2.1

<table>
<thead>
<tr>
<th>Value ID</th>
<th>Dictionary Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>apple</td>
</tr>
<tr>
<td>1</td>
<td>charlie</td>
</tr>
<tr>
<td>2</td>
<td>delta</td>
</tr>
<tr>
<td>3</td>
<td>frank</td>
</tr>
<tr>
<td>4</td>
<td>hotel</td>
</tr>
<tr>
<td>5</td>
<td>inbox</td>
</tr>
</tbody>
</table>

#### Outputs

**\( V^j_M \)**
- Value ID

**\( U^j_M \)**
- Dictionary Entries

**\( I^j \)**
- Postings offset

**\( U'^j_M \)**
- Postings

**\( I''_j \)**
- Read offset

**\( X^j_M \)**
- New Value-id

**\( D^j \)**
- Dictionary Entries

**\( P'^j \)**
- Position in Attribute Vector

**\( P^j \)**
- Position in Attribute Vector

---

**\( \mathbf{C} \) (corresponding tuples) = 0 1 3 5 7 9 11 12**

---

**Martin Faust | ADMS 2012**

---

**Monday, August 27, 12**
Extra Memory Costs

- Maintaining the index increases costs
- The old index has to be read and the new index has to be written
Offset Increased Memory Costs

- Index decreases memory traffic during query execution by turning column scans into lookups
- Index maintenance increases memory traffic of merge process
- Break-Even point depends on column parameters
- Savings of how many queries offset the increased merge costs?
Index Viability

Single Value Predicate: Index Savings vs. Maintenance

\[
\frac{N'_N}{N_M} = 1.01, N_M = 100M \quad \frac{N'_N}{N_M} = 1.01, N_M = 100K \quad \frac{N'_N}{N_M} = 1.01, N_M = 5K \\
\frac{N'_N}{N_M} = 1.10, N_M = 100M \quad \frac{N'_N}{N_M} = 1.10, N_M = 100K \quad \frac{N'_N}{N_M} = 1.10, N_M = 5K \\
\frac{N'_N}{N_M} = 1.50, N_M = 100M \quad \frac{N'_N}{N_M} = 1.50, N_M = 100K \quad \frac{N'_N}{N_M} = 1.50, N_M = 5K
\]

**Martin Faust | ADMS 2012**
Evaluation

Y1-Axis
- Create Index After Merge $N_M = 25M, N_D = 2.5M$
- Merge Column and Index $N_M = 25M, N_D = 2.5M$
- Merge w/o Index $N_M = 25M, N_D = 2.5M$

CPU Cycles

Distinct Value Fraction

$\times 10^{10}$
Conclusions

- Real-world measured CPU cycles are not only influenced by the memory traffic
- Increased code size
- Usage of write-combining buffers
- Memory traffic analysis allows to compare algorithms theoretically
- The general trend can be deducted from the analysis
Thank you!

Martin Faust
martin.faust@hpi.uni-potsdam.de