Adaptive String Dictionary Compression in In-Memory Column-Store Database Systems

Ingo Müller, Cornelius Ratsch, Franz Faerber / KIT / SAP AG
March 26, 2014 – EDBT, Athens, Greece
Motivation: Column-Store Architecture Recap

Logical representation

<table>
<thead>
<tr>
<th>First name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helen</td>
<td>2</td>
</tr>
<tr>
<td>Michael</td>
<td>3</td>
</tr>
<tr>
<td>Michael</td>
<td>3</td>
</tr>
<tr>
<td>Adam</td>
<td>1</td>
</tr>
<tr>
<td>Michael</td>
<td>3</td>
</tr>
<tr>
<td>Helen</td>
<td>2</td>
</tr>
<tr>
<td>Adam</td>
<td>1</td>
</tr>
</tbody>
</table>

Physical representation

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adam</td>
</tr>
<tr>
<td>2</td>
<td>Helen</td>
</tr>
<tr>
<td>3</td>
<td>Michael</td>
</tr>
</tbody>
</table>

Dictionary

- Focus on static dictionaries of the read-optimized store
Motivation: Observation of Real-World Data

- String columns play an important role in real-world applications
  → potential for compression
Outline

1. Introduction
2. Survey of Dictionary Formats
3. Performance Models
4. Automatic Selection
5. Evaluation
6. Summary
Survey of Dictionary Formats

Two basic dictionary formats

- Array
- Front coding

Combined with string compression schemes

- Uncompressed
- N-gram compression
- Bit compression
- Huffman
- Re-Pair
Formats provide trade-off between runtime and space consumption

Trade-off depends on dictionary content
Compression Models

Example: array of Huffman encoded strings

- size = |data| + # strings · |pointer|
- data = |raw data| · entropy₀

General idea: break down size into values that

- are either **known** or
- can be **sampled**
Compression Models: Evaluation

Compression models offer cheap, but accurate enough size predictions

max error = 5
Automatic Dictionary Selection: Goals and Overview

- Dictionary format
  - Compression model
  - Extract runtime per string
  - Locate runtime per string
  - Construct runtime per string

- Column
  - Number of extracts
  - Number of locates
  - Column vector size
  - Merge frequency
  - Dictionary content

- Database system
  - Occupied memory
  - Available memory

- Runtime in dictionary

- Dictionary size

- Selection strategy using $\Delta c$

- Dictionary selection

© 2014 SAP AG. All rights reserved.
Our heuristic selects a dictionary format based on local information and a global trade-off parameter ($\Delta c$).
Our heuristic selects a dictionary format based on local information and a global trade-off parameter ($\Delta c$).
Evaluation

Setup

- TPC-H with *key columns as strings \(\sim 47\) dictionaries
- Workload: all queries consecutively

Workload-driven dictionary selection outperforms static selection
- \(\Delta c\) effectively controls space / time trade-off
Summary

- Survey of dictionary implementations
  - Variety of space/time trade-offs depending on content
- Compression models
  - Feasible size prediction for assisted or automatic selection
- Automatic selection strategy
  - Heuristic translating a global trade-off parameter into local decisions
- Result
  - Workload-driven format selection improves overall system trade-off

Thank You!