Balanced Database Query Processing Based on Compressed Intermediates

Patrick Damme, Dirk Habich, Wolfgang Lehner

RoSI seminar, May 2, 2018
Importance of Analytical Queries

Analytical workloads

- Online Analytical Processing (OLAP)
  Workloads on growing data volumes
- Important in many fields
  (e.g. Business Intelligence)
Importance of Analytical Queries

Analytical workloads

- Online Analytical Processing (OLAP)
  Workloads on growing data volumes
- Important in many fields
  (e.g. Business Intelligence)
- Example (SQL, Star Schema Benchmark Q3)

```sql
SELECT c_nation, s_nation, d_year, SUM(lo_revenue) AS revenue
FROM customer, lineorder, supplier, date
WHERE lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_orderdate = d_datekey
  AND c_region = 'ASIA' AND s_region = 'ASIA'
  AND d_year >= 1992 AND d_year <= 1997
GROUP BY c_nation, s_nation, d_year
ORDER BY d_year ASC, revenue DESC;
```
Importance of Analytical Queries

Analytical workloads

- Online Analytical Processing (OLAP)
  Workloads on growing data volumes
- Important in many fields (e.g. Business Intelligence)
- Example (prose)

We want the total revenue generated by all orders posted by a customer from Asia and delivered by a supplier from Asia between 1992 and 1997 for all distinct combinations of the customer’s nation, the supplier’s nation, and the year sorted chronologically and in decreasing order of revenue
Importance of Analytical Queries

Analytical workloads

- Online Analytical Processing (OLAP)
  Workloads on growing data volumes
- Important in many fields (e.g. Business Intelligence)
- Example (prose)

We want the total revenue generated by all orders posted by a customer from Asia and delivered by a supplier from Asia between 1992 and 1997 for all distinct combinations of the customer’s nation, the supplier’s nation, and the year sorted chronologically and in decreasing order of revenue

Key requirement: Efficiency

- User wants the results quickly
- Optimum: interactive/"immediate" response
- Only possible, if DBMS is tailored to the underlying hardware
Importance of Analytical Queries

Analytical workloads

- Online Analytical Processing (OLAP)
  Workloads on growing data volumes
- Important in many fields (e.g. Business Intelligence)
- Example (prose)

We want the total revenue generated by all orders posted by a customer from Asia and delivered by a supplier from Asia between 1992 and 1997 for all distinct combinations of the customer’s nation, the supplier’s nation, and the year sorted chronologically and in decreasing order of revenue

Key requirement: Efficiency

- User wants the results quickly
- Optimum: interactive/"immediate" response
- Only possible, if DBMS is tailored to the underlying hardware

Classical, disk-centric DBMS
Importance of Analytical Queries

Analytical workloads
- Online Analytical Processing (OLAP)
  Workloads on growing data volumes
- Important in many fields (e.g. Business Intelligence)
- Example (prose)

We want the total revenue generated by all orders posted by a customer from Asia and delivered by a supplier from Asia between 1992 and 1997 for all distinct combinations of the customer’s nation, the supplier’s nation, and the year sorted chronologically and in decreasing order of revenue

Key requirement: Efficiency
- User wants the results quickly
- Optimum: interactive/"immediate" response
- Only possible, if DBMS is tailored to the underlying hardware

Nowadays’ in-memory DBMS
OLAP Workloads in In-Memory DBMSs
OLAP Workloads in In-Memory DBMSs
OLAP Workloads in In-Memory DBMSs

Bottleneck

- increasingly fast CPUs
- comparably low main memory bandwidth

CPU

Main Memory

Data
OLAP Workloads in In-Memory DBMSs

Bottleneck

- increasingly fast CPUs
- comparably low main memory bandwidth

Main Memory

Data

CPU

Columnar Layout
OLAP Workloads in In-Memory DBMSs

Bottleneck:
- Increasingly fast CPUs
- Comparably low main memory bandwidth

Recap: Row-Store vs. Column-Store

<table>
<thead>
<tr>
<th>OrderKey</th>
<th>PartKey</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>123</td>
<td>10</td>
</tr>
<tr>
<td>1002</td>
<td>45</td>
<td>20</td>
</tr>
<tr>
<td>1003</td>
<td>6789</td>
<td>15</td>
</tr>
<tr>
<td>1004</td>
<td>45</td>
<td>120</td>
</tr>
</tbody>
</table>
OLAP Workloads in In-Memory DBMSs

Recap: Row-Store vs. Column-Store

<table>
<thead>
<tr>
<th>OrderKey</th>
<th>PartKey</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>123</td>
<td>10</td>
</tr>
<tr>
<td>1002</td>
<td>45</td>
<td>20</td>
</tr>
<tr>
<td>1003</td>
<td>6789</td>
<td>15</td>
</tr>
<tr>
<td>1004</td>
<td>45</td>
<td>120</td>
</tr>
</tbody>
</table>
OLAP Workloads in In-Memory DBMSs

Bottleneck
increasingly fast CPUs
comparably low main memory bandwidth

Recap: Row-Store vs. Column-Store

<table>
<thead>
<tr>
<th>OrderKey</th>
<th>PartKey</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>123</td>
<td>10</td>
</tr>
<tr>
<td>1002</td>
<td>45</td>
<td>20</td>
</tr>
<tr>
<td>1003</td>
<td>6789</td>
<td>15</td>
</tr>
<tr>
<td>1004</td>
<td>45</td>
<td>120</td>
</tr>
</tbody>
</table>
OLAP Workloads in In-Memory DBMSs

**Bottleneck**
- increasingly fast CPUs
- comparably low main memory bandwidth

**Main Memory**
- Data

**Columnar Layout**
- Access only column relevant to the query
OLAP Workloads in In-Memory DBMSs

Bottleneck
- increasingly fast CPUs
- comparably low main memory bandwidth

Main Memory

Data

CPU

Columnar Layout
- Access only column relevant to the query

Lightweight data compression
- Reduced data sizes → lower transfer times
- → better cache utilization
- → fast direct processing of compressed data

Computational overhead
OLAP Workloads in In-Memory DBMSs

Bottleneck
-越来越快的CPU
-相对较低的内存带宽

CPU

Main Memory
- base data
- intermediates

Columnar Layout
- 只访问查询相关的列

Lightweight data compression
- 减少数据大小
  → 降低传输时间
  → 更好的缓存利用率
  → 快速直接处理压缩数据

Computational overhead
OLAP Workloads in In-Memory DBMSs

- **Bottleneck**
  - Increasingly fast CPUs
  - Comparably low main memory bandwidth

- **Main Memory**
  - Base data
  - Intermediates

- **Equal costs**
  - For accessing base data and intermediates

- **CPU**
  - Data access flow

- **Columnar Layout**
  - Access only column relevant to the query

- **Lightweight data compression**
  - Reduced data sizes
    - Lower transfer times
    - Better cache utilization
    - Fast direct processing of compressed data

- **Treat intermediates efficiently too!**
Our Vision: Balanced Query Processing based on Compressed Intermediate Results
Our Vision

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results
Our Vision

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

- Process compressed data directly
- Lightweight compression of all intermediate result
Our Vision:

- Benefit vs. overhead, optimize overall query performance
- Process compressed data directly
- Lightweight compression of all intermediate result

Our Vision: *Balanced* Query Processing *based on* Compressed Intermediate Results
Our Vision

Benefit vs. overhead, optimize overall query performance
Process compressed data directly
Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Structural Aspect

Investigation of lightweight compression
Our Vision

- Benefit vs. overhead, optimize overall query performance
- Process compressed data directly
- Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Structural Aspect
- Investigation of lightweight compression

Operational Aspect
- Integration of compression into query execution
Our Vision

Benefits vs. overhead, optimize overall query performance
Process compressed data directly
Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Structural Aspect
Investigation of lightweight compression

Operational Aspect
Integration of compression into query execution

Optimization Aspect
Compression-aware query optimization
Our Vision

Our Vision: **Balanced Query Processing** based on **Compressed Intermediate Results**

**Structural Aspect**
- Investigation of lightweight compression

**Operational Aspect**
- Integration of compression into query execution

**Optimization Aspect**
- Compression-aware query optimization

Completed work
Lossless Data Compression

**FOCUS ON INTEGERS**
- Due to their outstanding importance in column-stores
- Many columns contain integers by nature
- Other fixed-width types: representable as integers
- Variable-width types: representation by integer codes required for efficient processing
Lossless Data Compression

- Compression rates close to data’s entropy
- Slow → suitable for disk-centric systems
Lossless Data Compression

- Compression rates close to data’s entropy
- Slow → suitable for disk-centric systems

Huffman, LZW, LZ77

LZ78, Arithmetic Coding

Heavyweight Data Compression

DELTA, DICT, RLE

FOR, NS

Lightweight Data Compression

- Good compression rates
- Fast → suitable for in-memory systems
## Lightweight Data Compression Techniques

<table>
<thead>
<tr>
<th>RLE</th>
<th>Run Length Encoding</th>
<th>Replace run by value &amp; length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>1200</td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>300</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DELTA</th>
<th>Differential Coding</th>
<th>Replace data elem. by difference to predecessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>1150</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>1350</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>1355</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FOR</th>
<th>Frame-of-Reference Coding</th>
<th>Replace data elem. by difference to reference value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1050</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DICT</th>
<th>Dictionary Coding</th>
<th>Replace data elem. by 0-based key in dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1050</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1050</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NS</th>
<th>Null Suppression</th>
<th>Eliminate leading zeroes in binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example for Null Suppression**

00...001011

Example output: 1011
### Lightweight Data Compression Techniques

<table>
<thead>
<tr>
<th>RLE</th>
<th>DELTA</th>
<th>FOR</th>
<th>DICT</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Length Encoding</td>
<td>Differential Coding</td>
<td>Frame-of-Reference</td>
<td>Dictionary Coding</td>
<td>Null Suppression</td>
</tr>
<tr>
<td>Replace run by value &amp; length</td>
<td>Replace data elem. by difference to predecessor</td>
<td>Replace data elem. by difference to reference value</td>
<td>Replace data elem. by 0-based key in dictionary</td>
<td>Eliminate leading zeroes in binary representation</td>
</tr>
<tr>
<td>1200 1200 4 300 2 300</td>
<td>1000 1000 1100 50 1350 200 1355 5</td>
<td>1200 200 1100 100 1000 0 1050 50</td>
<td>1000 0 1200 1 1000 0 1050 2 1050 2</td>
<td>00...001011</td>
</tr>
</tbody>
</table>

Tailored to data characteristics
- compression rate
- performance
Lightweight Data Compression Techniques

<table>
<thead>
<tr>
<th>RLE</th>
<th>DELTA</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Length Encoding</td>
<td>Differential Coding</td>
<td>Null Suppression</td>
</tr>
<tr>
<td>Replace run by value &amp; length</td>
<td>Replace data element by difference to predecessor</td>
<td>Eliminate leading zeroes in binary representation</td>
</tr>
<tr>
<td>1200</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1200</td>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1200</td>
<td>1100</td>
<td>0</td>
</tr>
<tr>
<td>1200</td>
<td>1150</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>1350</td>
<td>50</td>
</tr>
<tr>
<td>300</td>
<td>1355</td>
<td>5</td>
</tr>
<tr>
<td>300</td>
<td>1200</td>
<td>0</td>
</tr>
<tr>
<td>1200</td>
<td>1100</td>
<td>1</td>
</tr>
<tr>
<td>1200</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1050</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1050</td>
<td>2</td>
</tr>
</tbody>
</table>

Tailored to data characteristics
determining their
- compression rate
- performance

High number of algorithms
- many for each technique
- even more by combinations
# Lightweight Data Compression Techniques

## RLE
- **Run Length Encoding**
- Replace run by value & length

<table>
<thead>
<tr>
<th>Value</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>1200</td>
<td>4</td>
</tr>
<tr>
<td>1200</td>
<td>300</td>
</tr>
<tr>
<td>1200</td>
<td>2</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
</tr>
</tbody>
</table>

## DELTA
- **Differential Coding**
- Replace data elem. by difference to predecessor

<table>
<thead>
<tr>
<th>Value</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>1100</td>
<td>100</td>
</tr>
<tr>
<td>1150</td>
<td>50</td>
</tr>
<tr>
<td>1350</td>
<td>200</td>
</tr>
<tr>
<td>1355</td>
<td>5</td>
</tr>
<tr>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>1100</td>
<td>100</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1050</td>
<td>50</td>
</tr>
</tbody>
</table>

## FOR
- **Frame-of-Reference**
- Replace data elem. by difference to reference value

<table>
<thead>
<tr>
<th>Value</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>200</td>
</tr>
<tr>
<td>1100</td>
<td>100</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1050</td>
<td>50</td>
</tr>
</tbody>
</table>

## DICT
- **Dictionary Coding**
- Replace data elem. by 0-based key in dictionary

<table>
<thead>
<tr>
<th>Value</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1200</td>
<td>1</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1050</td>
<td>2</td>
</tr>
</tbody>
</table>

## NS
- **Null Suppression**
- Eliminate leading zeroes in binary representation

<table>
<thead>
<tr>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00...001011</td>
</tr>
</tbody>
</table>

### Tailored to data characteristics
- determining their compression rate
- performance

### High number of algorithms
- many for each technique
- even more by combinations

### Efficiency is crucial for us
- (de)compression during query execution
- use of SIMD extensions
A Sample of Our Experimental Survey
A Sample of Our Experimental Survey

Compression rate [bits/int]

Compression time [sec]

Decompression time [sec]

Algorithm A
Algorithm B
Algorithm C
A Sample of Our Experimental Survey

There is no single-best compression algorithm. The choice is non-trivial and depends on:
There is no single-best compression algorithm

The choice is non-trivial and depends on:

Data characteristics
A Sample of Our Experimental Survey

There is no single-best compression algorithm. The choice is non-trivial and depends on:

- Data characteristics
- Optimization target
A Sample of Our Experimental Survey

Compression rate [bits/int]

Compression time [sec]

Decompression time [sec]

Dataset 1  Dataset 2  Dataset 3

Algorithm A  Algorithm B  Algorithm C

Missing the right choice can lead to
- performance decrease of factors
- memory consumption increase of factors
Changing Data Characteristics

Base data

INSERT

UPDATE

DELETE

t_1

t_2

data distribution

frequency

value

time
Changing Data Characteristics

Base data

INSERT
UPDATE
DELETE

t₁

data distribution

frequency

value

t₂

data distribution

frequency

value

time
Changing Data Characteristics

- INSERT
- UPDATE
- DELETE

Base data

We might need to change the format efficiently

And this should be done efficiently

Data distribution at time $t_1$

Data distribution at time $t_2$

value

frequency
Changing Data Characteristics
Changing Data Characteristics

Base data

Data distribution

Frequency

Value

Base data

B

A

C

2

1

3
Changing Data Characteristics

- Base data
- Intermediate
- Selection

Data distribution:
- Frequency vs. Value

Values:
- A
- B
- C

Counts:
- 2
- 1
- 3
Changing Data Characteristics

- **Base data** selection
- **Intermediate** value frequency
- **Data distribution**

Diagram:
- Base data
- Selection
- Intermediate
- Value frequency
- Data distribution

Symbols:
- B
- A
- C
- 2
- 1
- 3
Changing Data Characteristics

- **Base Data**:
  - Data distribution
  - Frequency vs. Value

- **Intermediate**

- **Selection**
  - Data distribution
  - Frequency vs. Value

- **Example**
  - Values: A, B, C
  - Frequencies: 2, 1, 3
Changing Data Characteristics

- Base data
- Intermediate
- Selection

Data distribution

Value

Frequency
Changing Data Characteristics

Base data

Selection

Intermediate

Aggregation

Data distribution

Value

Frequency

Data distribution

Value

Frequency
Changing Data Characteristics

Base data

Selection

Intermediate

Aggregation

Data distribution

frequency

value

Data distribution

frequency

value

Base data

Total query execution time

Keeping format A

Changing to format C
Changing Data Characteristics

We might need to change the format and this must be done very efficiently.
Data Transformation Techniques

Data Transformation

Adapting the representation of the data from a source format to a destination format
Data Transformation Techniques

Data Transformation

Adapting the representation of the data from a source format to a destination format

Indirect transformation

Complete decompression and recompression

Compressed (src. format) → uncompressed → Compressed (dest. format)

Relies only on existing (de)compression algorithms
Inefficient materialization of the uncompressed data
Data Transformation Techniques

Indirect transformation

Complete decompression and recompression

uncompressed

compressed (src. format)

compressed (src. format)

Transformation in one step

Direct transformation

Relies only on existing (de)compression algorithms

Inefficient materialization of the uncompressed data

Novel class of algorithms related to (de)compression

Can be much more efficient depending on the data
Example: From RLE to 4-Wise NS
Example: From RLE to 4-Wise NS

Indirect transformation

main memory

RLE

12

main memory

4-Wise NS
Example: From RLE to 4-Wise NS

Indirect transformation

- RLE
- uncompr.
- 4-Wise NS
Example: From RLE to 4-Wise NS

**Indirect transformation**
- **main memory**
  - RLE
  - **uncompr.**
  - 12

**Direct transformation**
- **main memory**
  - RLE
  - **uncompr.** | 4-Wise NS
  - **CPU vector registers**
  - 4-Wise NS

4-Wise NS
**Example: From RLE to 4-Wise NS**

**Indirect transformation**

- **main memory**
  - RLE
  - **uncompr.**
  - 12

- **main memory**
  - 4-Wise NS

**Direct transformation**

- **main memory**
  - CPU vector registers
  - RLE
  - **uncompr. | 4-Wise NS**

- **main memory**
  - 4-Wise NS

---

*We save*
- Storing/loading uncompressed data to/from main memory
- Repeated block recompression
Our Vision

Our Vision: **Balanced Query Processing** based on **Compressed Intermediate Results**

**Structural Aspect**
- Investigation of lightweight compression

**Operational Aspect**
- Integration of compression into query execution

**Optimization Aspect**
- Compression-aware query optimization

Completed work
Our Vision: Balanced Query Processing based on Compressed Intermediate Results

- **Structural Aspect**
  - Investigation of lightweight compression

- **Operational Aspect**
  - Integration of compression into query execution

- **Optimization Aspect**
  - Compression-aware query optimization

**Completed work**

**Ongoing work**
Baseline: Uncompressed Processing

Inputs and outputs
- Each might be represented in its individual compressed format

We assume
- We want to support
  - $n$ compressed formats
  - 1 operator

OP$_U$
Inputs and outputs
- Each might be represented in its individual compressed format

We assume
- We want to support
  - $n$ compressed formats
  - 1 operator

Different degrees of integration possible
Wrapper Employing (De)Compression

Idea
- Wrapper around original operator
  - Decompresses inputs
  - Recompresses outputs
- Efficiency: work on blocks fitting into the L1-cache
- Format changes through wrapper configuration

Requires
- 1 operator variant
- $n$ compressions
- $n$ decompressions

Advantage
- Simplicity: relies only on existing
  - operator implementations
  - (de)compression algorithms

Disadvantage
- Wastes potential of working directly on compressed data
Wrapper Employing Direct Transformation

Idea
- Direct processing of compressed data elements
- Common format for all inputs and outputs
- Arbitrary format combinations through a wrapper employing our direct transformations

Requires
- \( n \) operator variants
- \( n^2 - n \) transformations

Advantage
- Compr. data within operators
  - Better cache utilization
  - Higher bit-level-parallelism when using SIMD instructions

Disadvantage
- Wrapper still causes overhead
Specialized Operator

Idea
- Operator tailored to a specific format combination
- No wrapper

Requires
- $n^{i+o}$ operator variants

Advantage
- Maximum efficiency

Disadvantage
- Highest integration effort due to high number of operator variants
Our Vision: Balanced Query Processing based on Compressed Intermediate Results

**Structural Aspect**
Investigation of lightweight compression

**Operational Aspect**
Integration of compression into query execution

**Optimization Aspect**
Compression-aware query optimization

Completed work

Ongoing work
Our Vision

Our Vision: **Balanced Query Processing** based on **Compressed Intermediate Results**

**Structural Aspect**
- Investigation of lightweight compression
  - Completed work

**Operational Aspect**
- Integration of compression into query execution
  - Ongoing work

**Optimization Aspect**
- Compression-aware query optimization
  - Future work
Compression-Aware Query Optimization
Compression-Aware Query Optimization

Existing query optimizer

SQL query

“optimal” plan
Compression-Aware Query Optimization

Existing query optimizer

- Algebraic restructuring
- Join ordering
- Logical ops to physical ops
- ...

SQL query

“optimal” plan
Compression-Aware Query Optimization

Existing query optimizer

- Algebraic restructuring
- Join ordering
- Logical ops to physical ops
- ...

Deep integration is beyond our scope.

“optimal” plan
Compression-Aware Query Optimization

Deep integration is beyond our scope.

Existing query optimizer

- Algebraic restructuring
- Join ordering
- Logical ops to physical ops
- ...

“optimal” plan w/o compression

Compression-aware optimization strategies

Our contribution

Our contribution

improved plan w/ compression

SQL query

SELECT ...

highly complex
Compression-Aware Query Optimization

Deep integration is beyond our scope.

Existing query optimizer

- Algebraic restructuring
- Join ordering
- Logical ops to physical ops
- ...

“optimal” plan w/o compression

Compression-aware optimization strategies

- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data

Our contribution
Challenges to Be Addressed

Compression-aware optimization strategies

- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data

Our contribution
Challenges to Be Addressed

Estimation of the intermediates' data characteristics

Our contribution

Compression-aware optimization strategies

- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data
Challenges to Be Addressed

- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data
Challenges to Be Addressed

Cost model for compression algorithms and operators

Our contribution

Compression-aware optimization strategies

- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data

improved plan w/ compression
Decision in Context

User side
- Run time requirements (interactive vs. batch)
- Importance of the query (e.g. who submitted it)

System side
- Current system load
- Optimize run time with limited memory consumption
- Optimize memory consumption with limited run time

Our contribution

Compression-aware optimization strategies

- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data

improved plan w/ compression
Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Structural Aspect
- Investigation of lightweight compression
  - Completed work

Operational Aspect
- Integration of compression into query execution
  - Ongoing work

Optimization Aspect
- Compression-aware query optimization
  - Future work
Summary and Contributions

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Contributions so far

- **Experimental survey** of lightweight compr. algos
  - EDBT'17 paper + demo
  - ACM TODS submission/revision

- **Direct transformations** for compressed data
  - ADBIS'15 paper

- **Benchmark framework** for lightweight compr. algos
  - TPCTC'15 paper
Summary and Contributions

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Contributions so far

- Experimental survey of lightweight compr. algos
- Direct transformations for compressed data
- Benchmark framework for lightweight compr. algos

Contributions still in the works

- EDBT’17 paper + demo ACM TODS submission/revision
- Operators for compressed data with different degrees of integration
- Cost model for lightweight compression algorithms and operators for compressed data
- Compression-aware strategies for the query optimizer

Our Vision: