TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

Balanced Query Processing Based on Lightweight Compression of Intermediates

Patrick Damme, 1st TAB presentation

Supervisor: Wolfgang Lehner
Co-supervisor: Thorsten Strufe
RoSI Post-Doc: Dirk Habich

Importance of Analytical Queries

Dresden Database Systems Group

Analytical workloads

- Growing data volumes in numerous areas
- Online Analytical Processing (OLAP) Workloads gain importance

Importance of Analytical Queries

Dresden Database Systems Group

Analytical workloads

- Growing data volumes in numerous areas
- Online Analytical Processing (OLAP) Workloads gain importance

- Star Schema Benchmark Query 3 (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

Dresden Database Systems Group

Analytical workloads

- Growing data volumes in numerous areas
- Online Analytical Processing (OLAP) Workloads gain importance

- Star Schema Benchmark Query 3 (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

Dresden Database Systems Group

Analytical workloads

- Growing data volumes in numerous areas
- Online Analytical Processing (OLAP) Workloads gain importance

- Star Schema Benchmark Query 3 (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

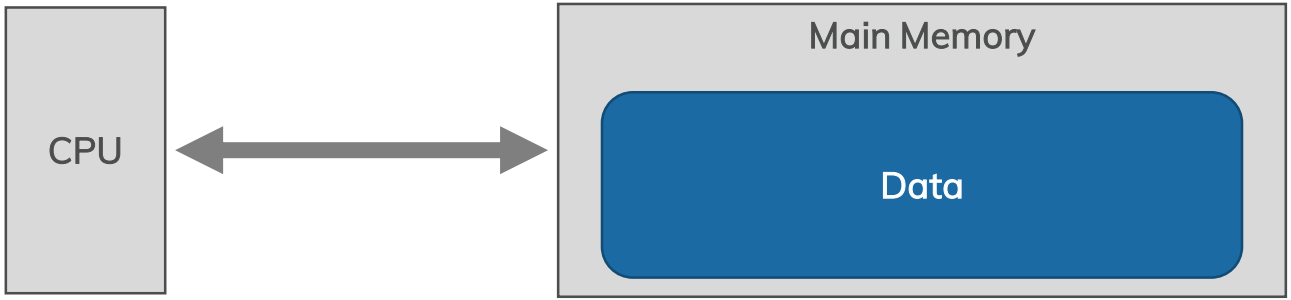
Column-Stores

- DB architecture of choice for OLAP workloads
- Store data elements of one column in subsequent memory locations
- Nowadays fully in-memory for efficiency

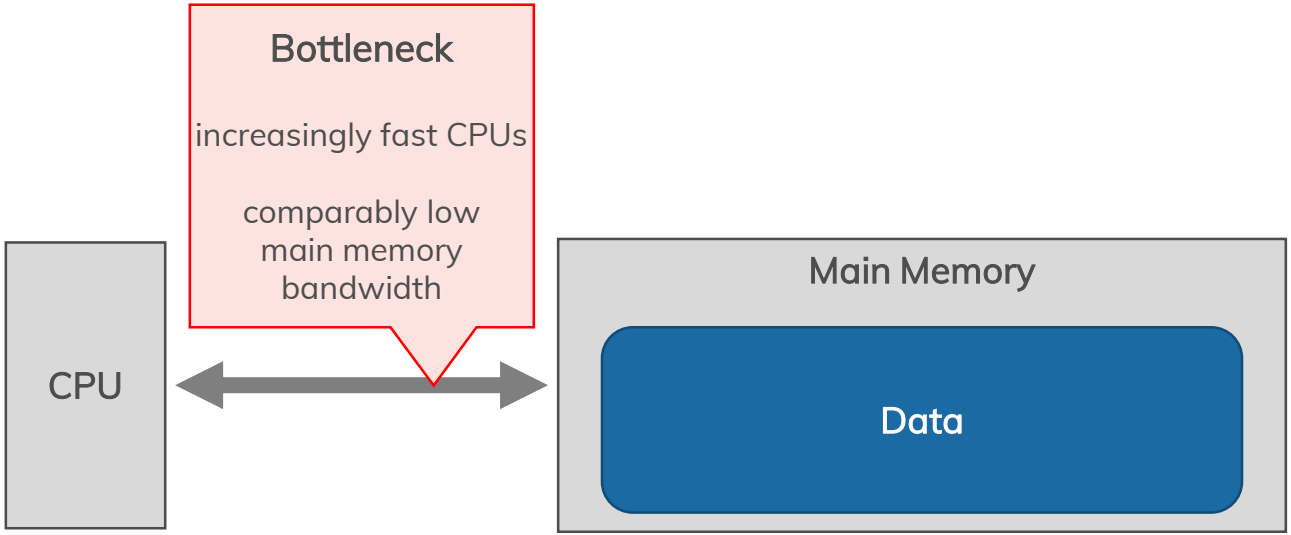
OLAP Workloads in In-Memory Column-Stores

Data

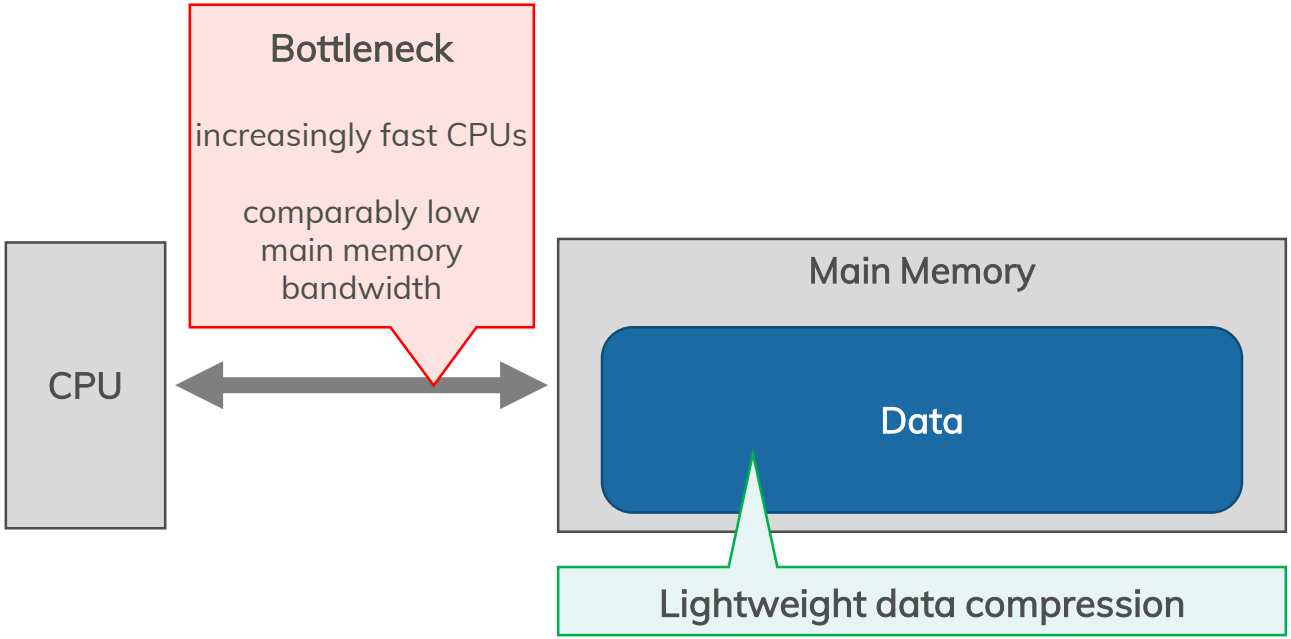
OLAP Workloads in In-Memory Column-Stores



OLAP Workloads in In-Memory Column-Stores



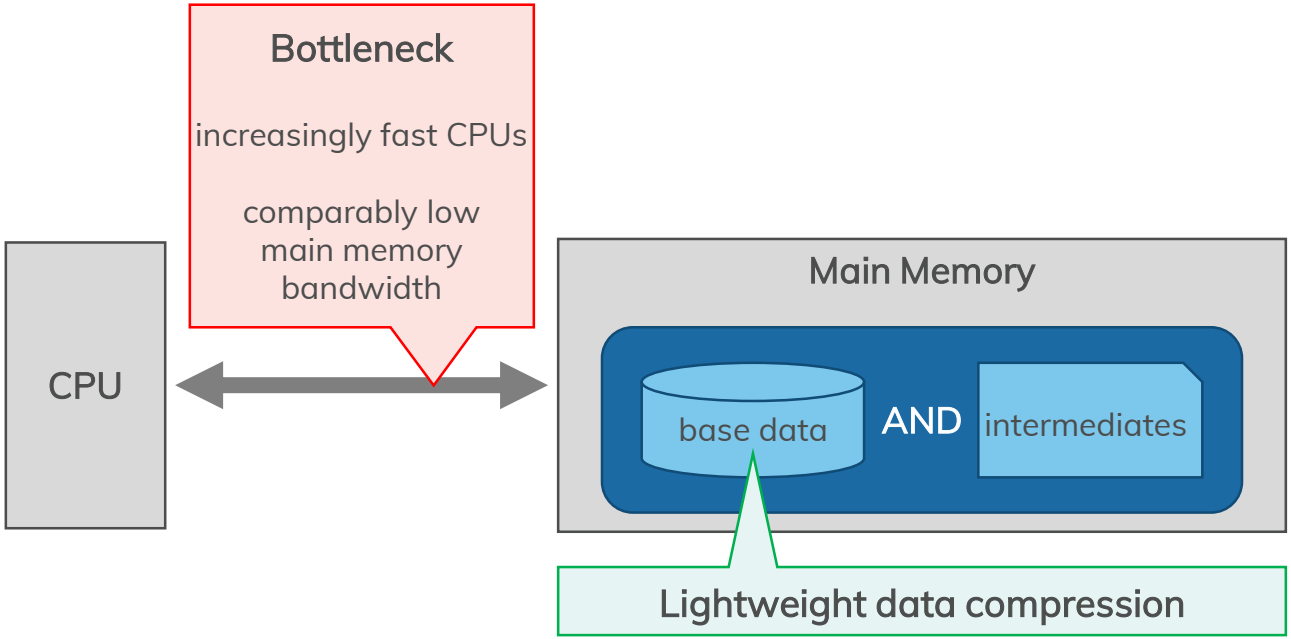
OLAP Workloads in In-Memory Column-Stores



- Reduced data sizes
  → lower transfer times
  → better cache utilization
- Direct processing of compressed data

- Computational overhead

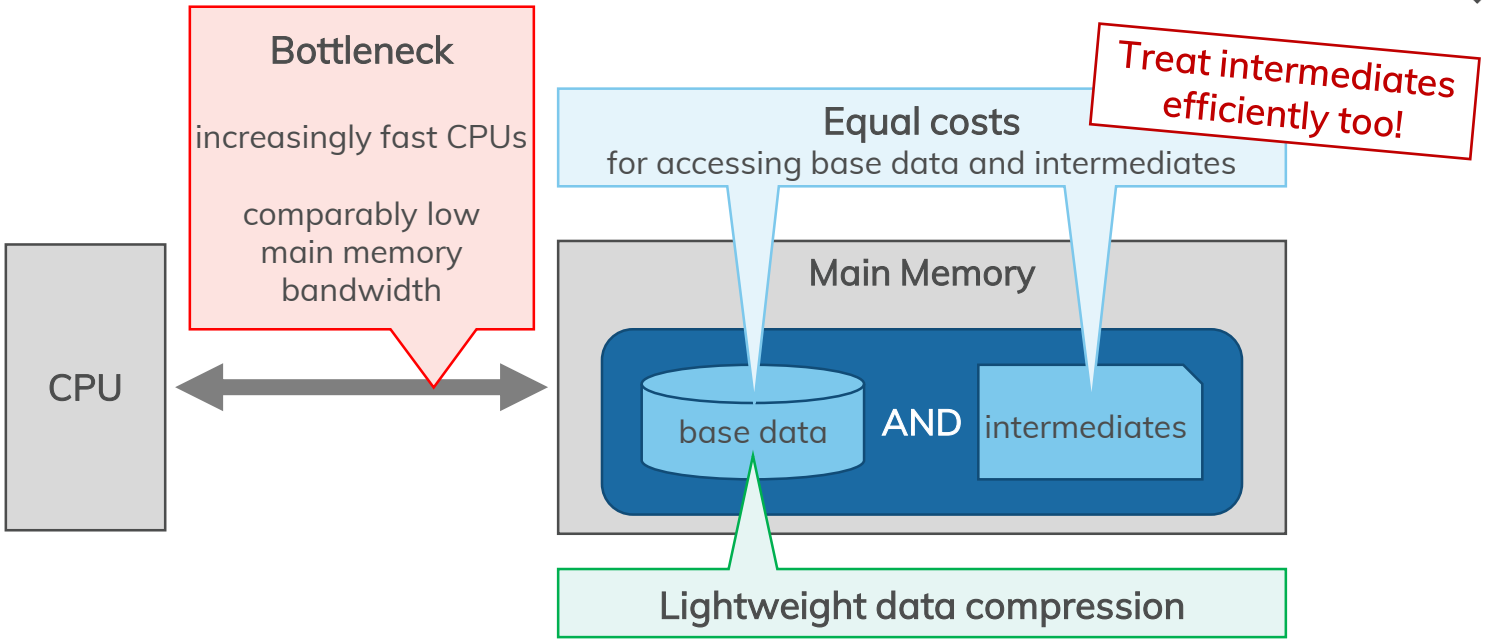
OLAP Workloads in In-Memory Column-Stores



- Reduced data sizes
  → lower transfer times
  → better cache utilization
- Direct processing of compressed data

- Computational overhead

OLAP Workloads in In-Memory Column-Stores

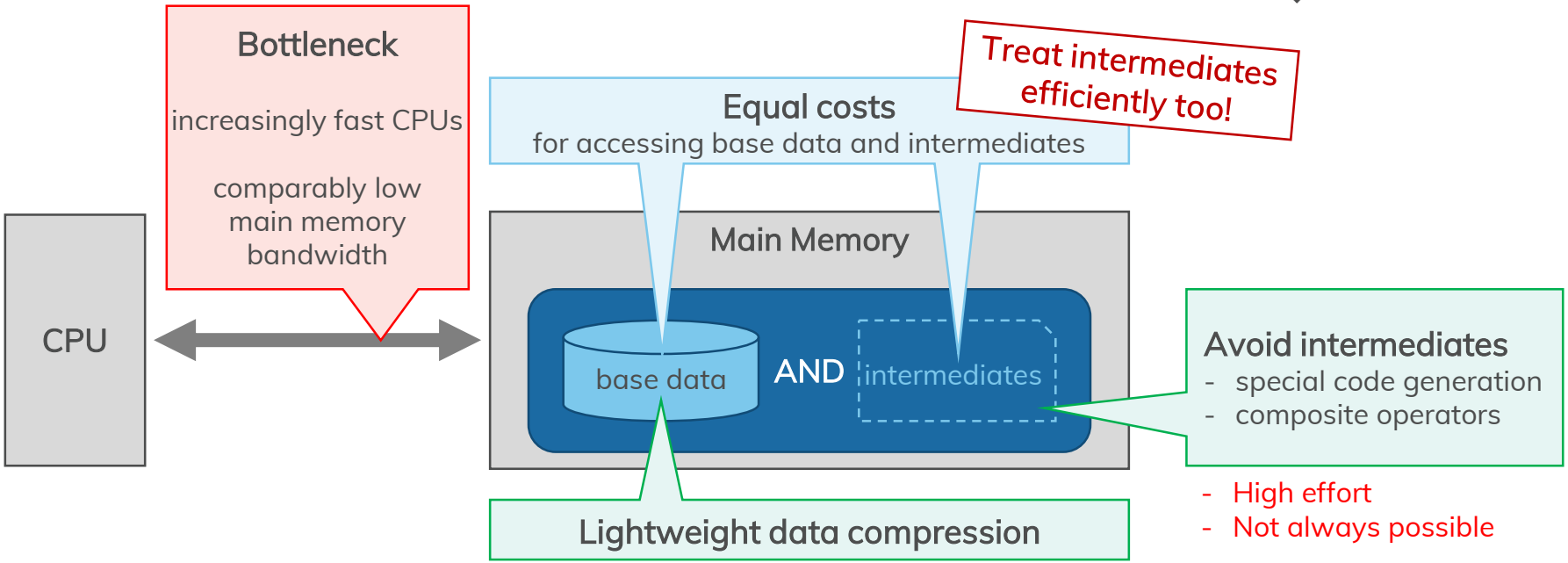


- Reduced data sizes
  → lower transfer times
  → better cache utilization
- Direct processing of compressed data

- Computational overhead

OLAP Workloads in In-Memory Column-Stores

Dresden Database Systems Group

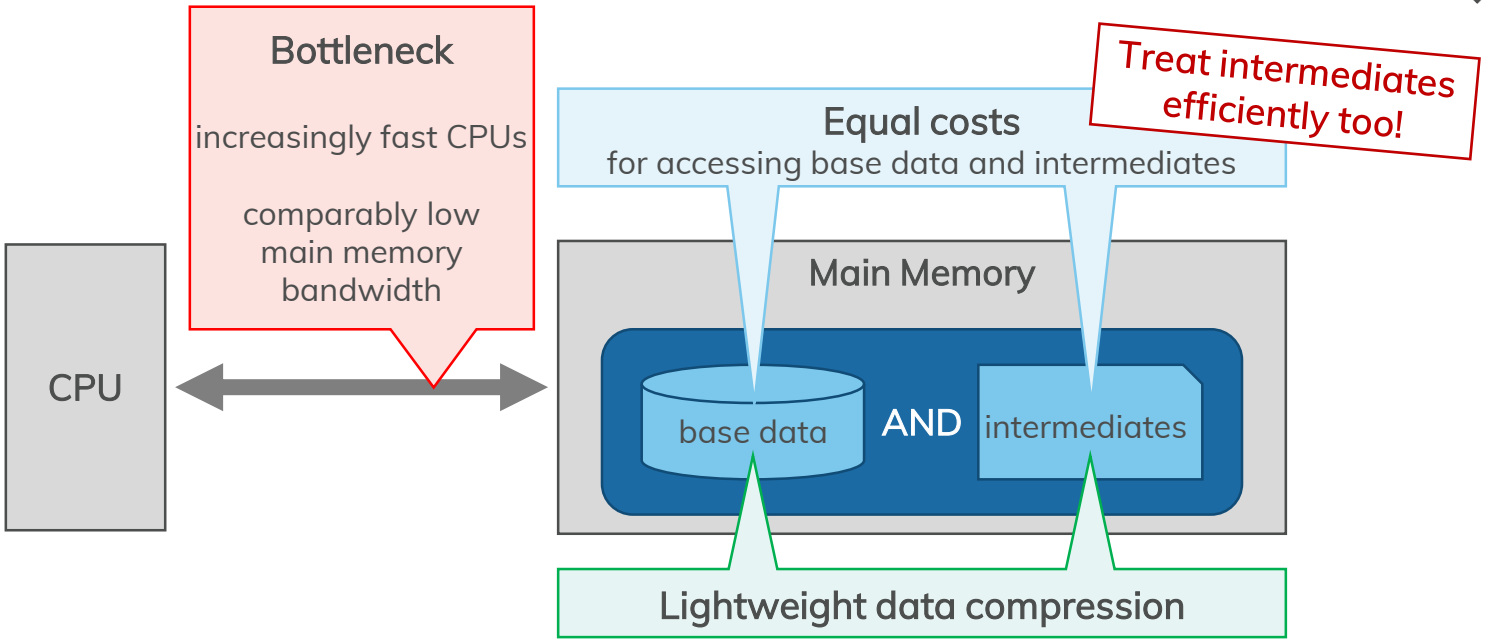


- High effort
- Not always possible

- Reduced data sizes
  → lower transfer times
  → better cache utilization
- Direct processing of compressed data

- Computational overhead

OLAP Workloads in In-Memory Column-Stores



- Reduced data sizes
  → lower transfer times
  → better cache utilization
- Direct processing of compressed data

- Computational overhead

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Our Vision

Dresden Database Systems Group

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Our Vision

Dresden Database Systems Group

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Our Vision

Dresden Database Systems Group

Benefit vs. overhead, optimize overall query performance

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Our Vision

Dresden Database Systems Group

Benefit vs. overhead, optimize overall query performance

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Our Vision

Dresden Database Systems Group

Benefit vs. overhead, optimize overall query performance

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics

Surrounding operators

System utilization, user preferences

Our Vision

Dresden Database Systems Group

Benefit vs. overhead, optimize overall query performance

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics

Structural Aspect

Investigation of lightweight compression

Surrounding operators

System utilization, user preferences

Our Vision

Dresden Database Systems Group

Benefit vs. overhead, optimize overall query performance

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	
Investigation of lightweight compression	Integration of compression into query execution	

Our Vision

Dresden Database Systems Group

Benefit vs. overhead, optimize overall query performance

Compression-aware physical operators

Lightweight compression of all intermediate result

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization

Completed work

Lightweight Data Compression Techniques

Dresden Database Systems Group

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

Lightweight Data Compression Techniques

Dresden Database Systems Group

RLE

Run Length Encoding
Replace run by value & length

1200		1200
1200		4
1200	▶	300
1200		2
300		
300		

Lightweight Data Compression Techniques

Dresden Database Systems Group

RLE	
Run Length Encoding Replace run by value & length	
1200 1200 1200 1200 300 300	1200 4 300 2

DELTA	
Differential Coding Replace data elem. by difference to predecessor	
1000 1100 1150 1350 1355	1000 100 50 200 5

Lightweight Data Compression Techniques

Dresden Database Systems Group

RLE	DELTA	FOR
Run Length Encoding Replace run by value & length	**Differential Coding** Replace data elem. by difference to predecessor	**Frame-of-Reference** Replace data elem. by difference to reference value
1200 1200 1200 1200 300 300 ▶ 1200 4 300 2	1000 1100 1150 1350 1355 ▶ 1000 100 50 200 5	1200 1100 1000 1050 ▶ 200 100 0 50

Lightweight Data Compression Techniques

Dresden Database Systems Group

RLE	DELTA	FOR	DICT
Run Length Encoding Replace run by value & length	**Differential Coding** Replace data elem. by difference to predecessor	**Frame-of-Reference** Replace data elem. by difference to reference value	**Dictionary Coding** Replace data elem. by 0-based key in dictionary
1200 → 1200 1200 → 4 1200 → 300 1200 → 2 300 300	1000 → 1000 1100 → 100 1150 → 50 1350 → 200 1355 → 5	1200 → 200 1100 → 100 1000 → 0 1050 → 50	1000 → 0 1200 → 1 1000 → 0 1050 → 2 1050 → 2

Lightweight Data Compression Techniques

Dresden Database Systems Group

RLE

Run Length Encoding
Replace run by value & length

1200	1200
1200	4
1200	300
1200	2
300	
300	

DELTA

Differential Coding
Replace data elem. by difference to predecessor

1000	1000
1100	100
1150	50
1350	200
1355	5

FOR

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

1200	200
1100	100
1000	0
1050	50

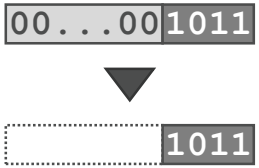
DICT

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

1000	0
1200	1
1000	0
1050	2
1050	2

NS

Null Suppression
Eliminate leading zeroes in binary representation



Lightweight Data Compression Techniques

Dresden Database Systems Group

RLE

Run Length Encoding
Replace run by value & length

1200	1200
1200	4
1200	300
1200	2
300	
300	

DELTA

Differential Coding
Replace data elem. by difference to predecessor

1000	1000
1100	100
1150	50
1350	200
1355	5

FOR

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

1200	200
1100	100
1000	0
1050	50

DICT

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

1000	0
1200	1
1000	0
1050	2
1050	2

NS

Null Suppression
Eliminate leading zeroes in binary representation

00...001011

1011

Tailored to data characteristics

determining their
- compression rate
- performance

Lightweight Data Compression Techniques

Dresden Database Systems Group

Varint-G8IU, PFOR, Masked-VByte, Simple-8b, PFOR2008, SIMD-GroupSimple, SIMD-BP128, SIMD-FastPFOR, Simple-16, Simple-9, VByte, 4-Wise NS, Varint-GB, 4-Gamma, NewPFOR, ...

RLE

Run Length Encoding
Replace run by value & length

1200	1200
1200	4
1200	300
1200	2
300	
300	

DELTA

Differential Co
Replace data e
by difference
predecessor

1000	1000
1100	100
1150	50
1350	200
1355	5

reference value

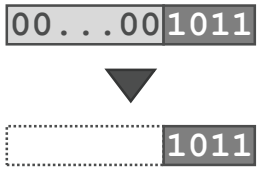
1200	200
1100	100
1000	0
1050	50

in dictionary

1000	0
1200	1
1000	0
1050	2
1050	2

NS

Null Suppression
Eliminate leading zeroes in binary representation



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

Dresden Database Systems Group

RLE

Run Length Encoding
Replace run by value & length

1200	1200
1200	4
1200	300
1200	2
300	
300	

DELTA

Differential Coding
Replace data elem. by difference to predecessor

1000	1000
1100	100
1150	50
1350	200
1355	5

FOR

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

1200	200
1100	100
1000	0
1050	50

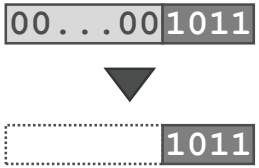
DICT

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

1000	0
1200	1
1000	0
1050	2
1050	2

NS

Null Suppression
Eliminate leading zeroes in binary representation



Tailored to data characteristics

determining their
- compression rate
- performance

High number of algorithms

- many for each technique
- even more by combinations

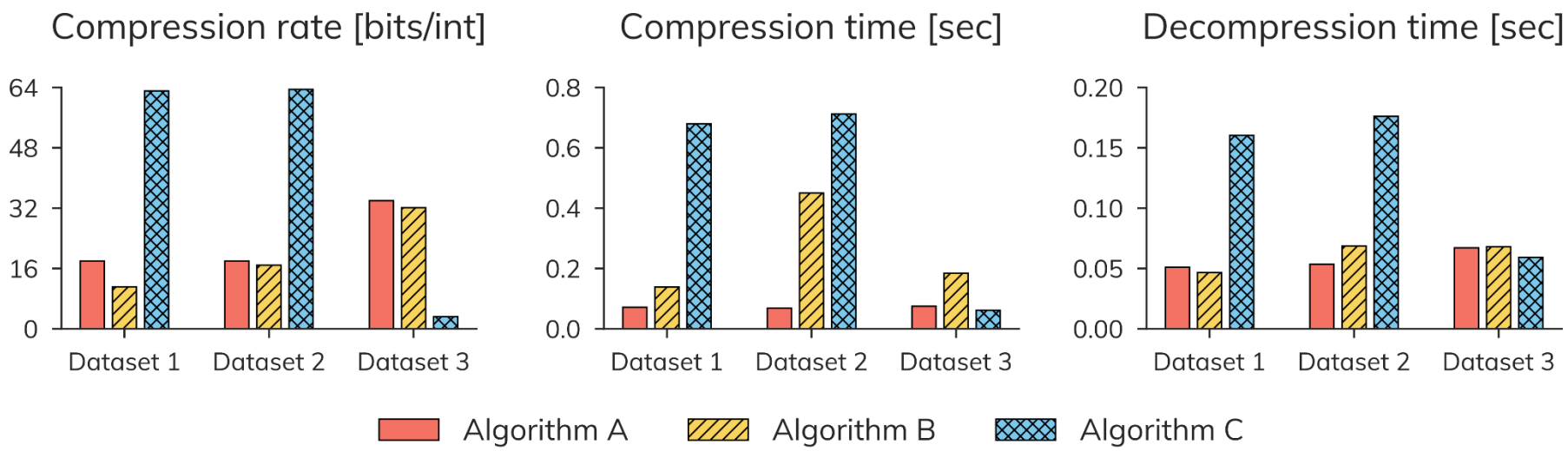
Efficiency is crucial for us

- (de)compression during query execution
- use of SIMD extensions

A Sample of Our Experimental Survey

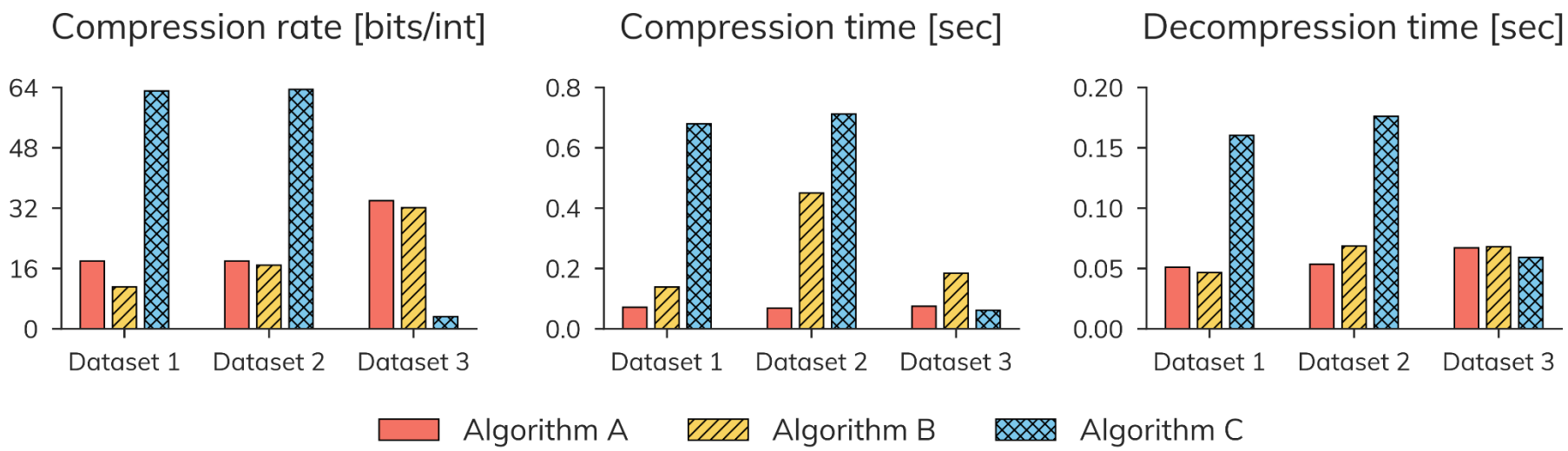
A Sample of Our Experimental Survey

Dresden Database Systems Group



A Sample of Our Experimental Survey

Dresden Database Systems Group

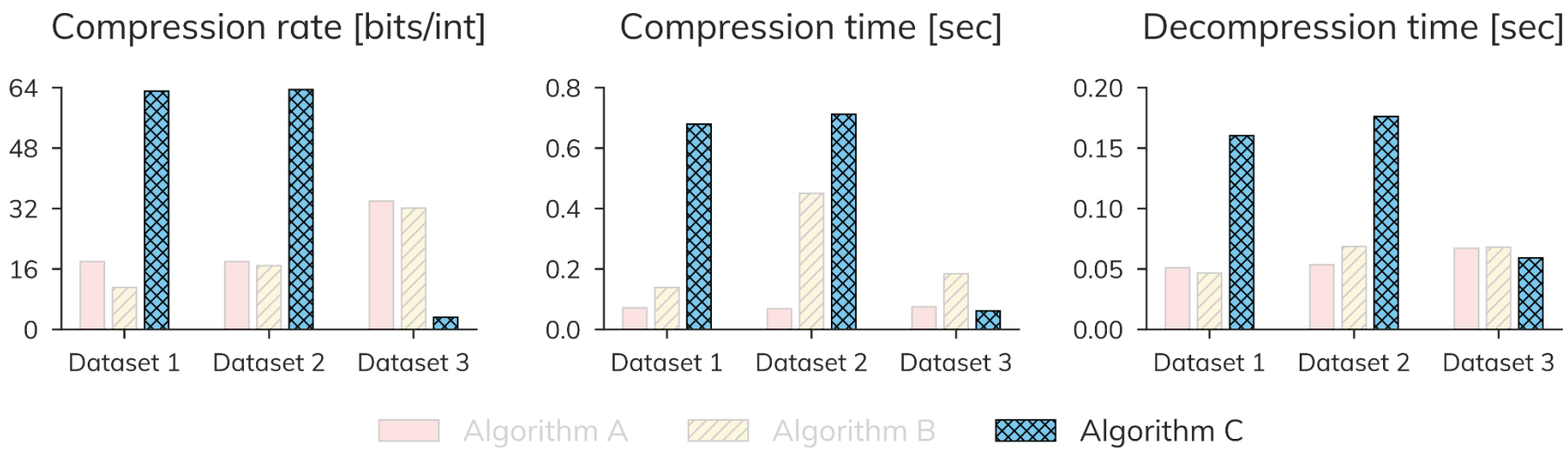


There is no single-best compression algorithm

The choice is non-trivial and depends on:

A Sample of Our Experimental Survey

Dresden Database Systems Group



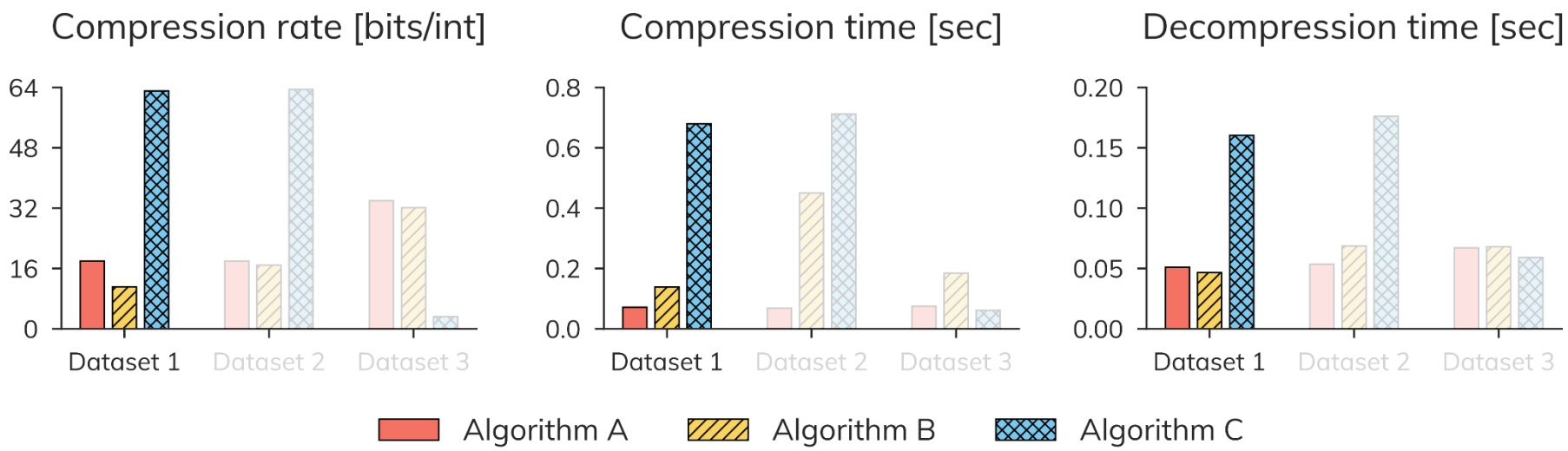
There is no single-best compression algorithm

The choice is non-trivial and depends on:

Data characteristics

A Sample of Our Experimental Survey

Dresden Database Systems Group



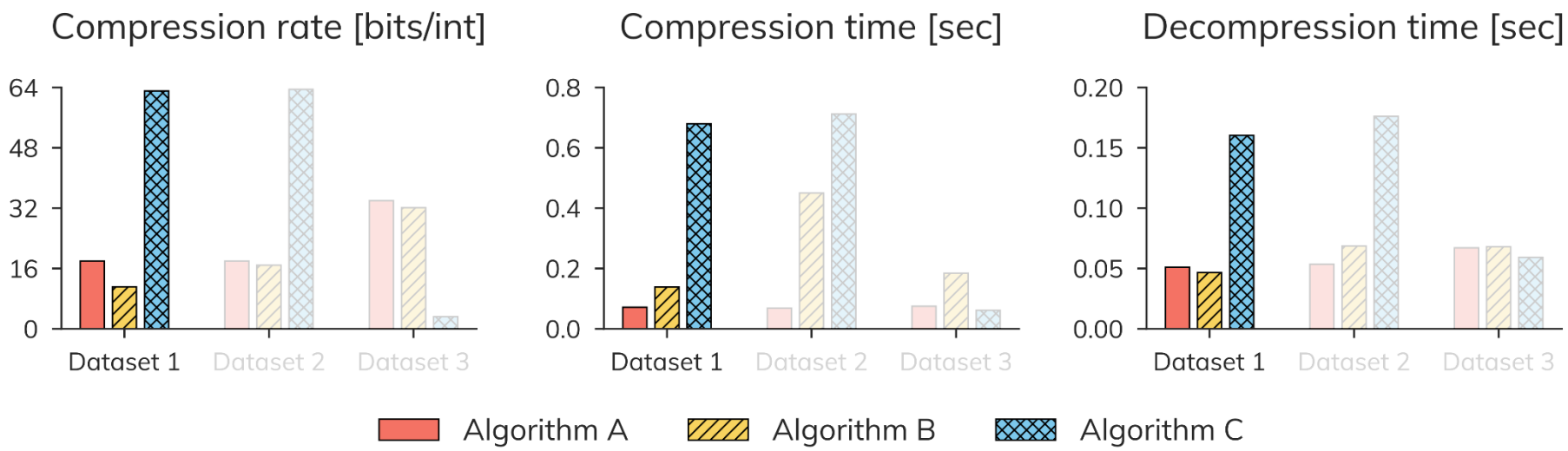
There is no single-best compression algorithm

The choice is non-trivial and depends on:

Data characteristics | Objective

A Sample of Our Experimental Survey

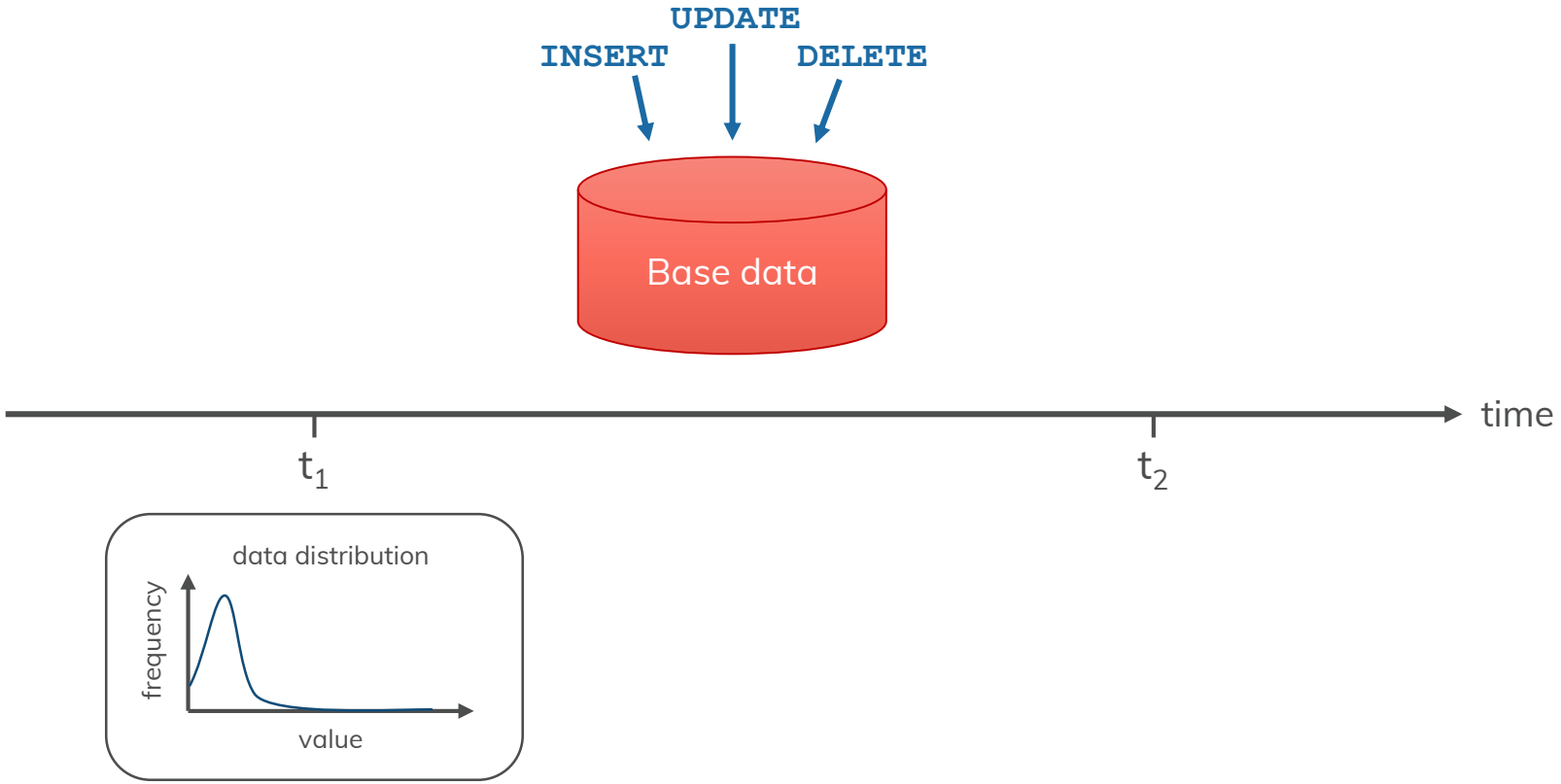
Dresden Database Systems Group



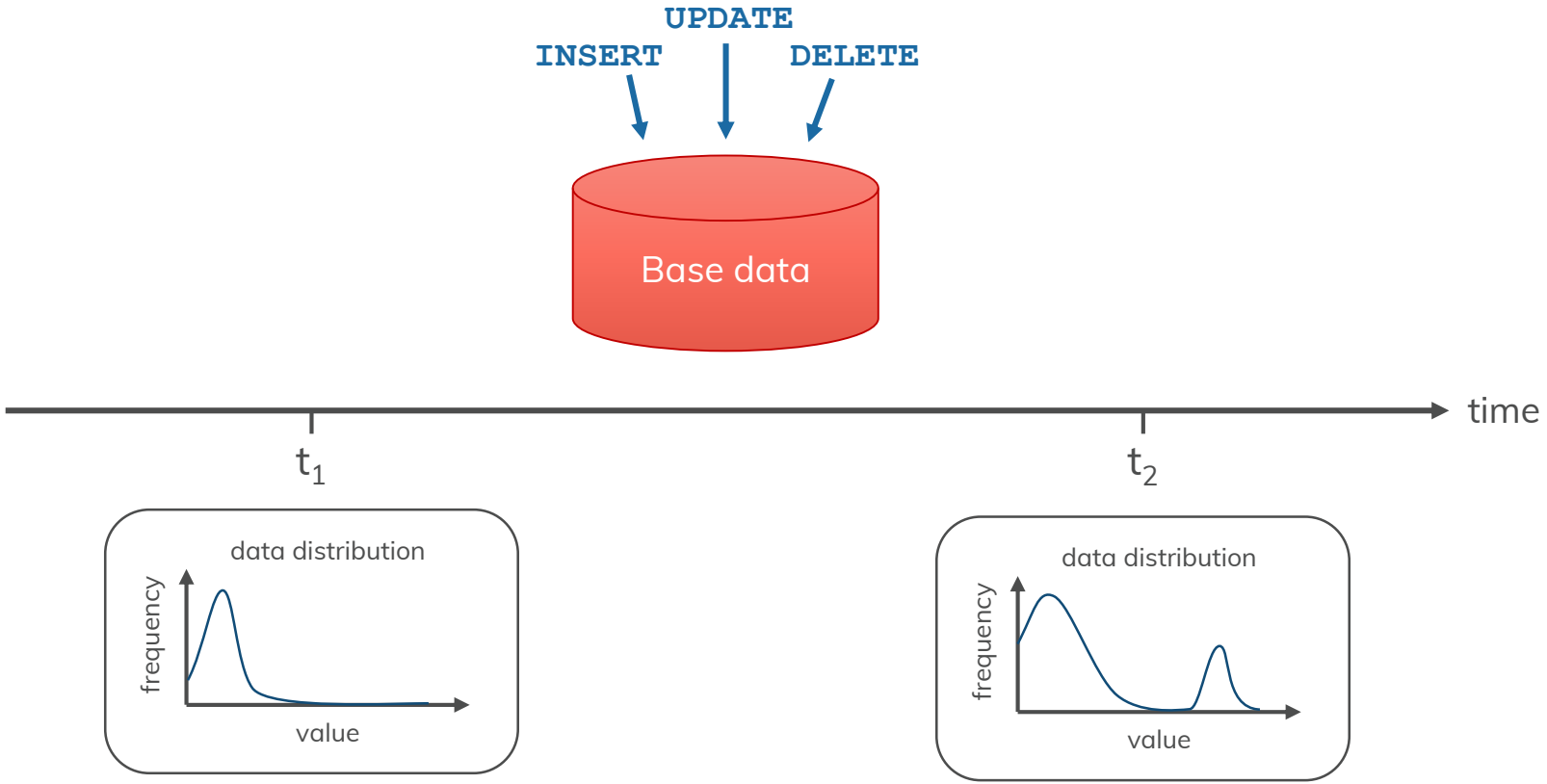
Missing the right choice can lead to
- performance decrease of factors
- memory consumption increase of factors

Changing Data Characteristics

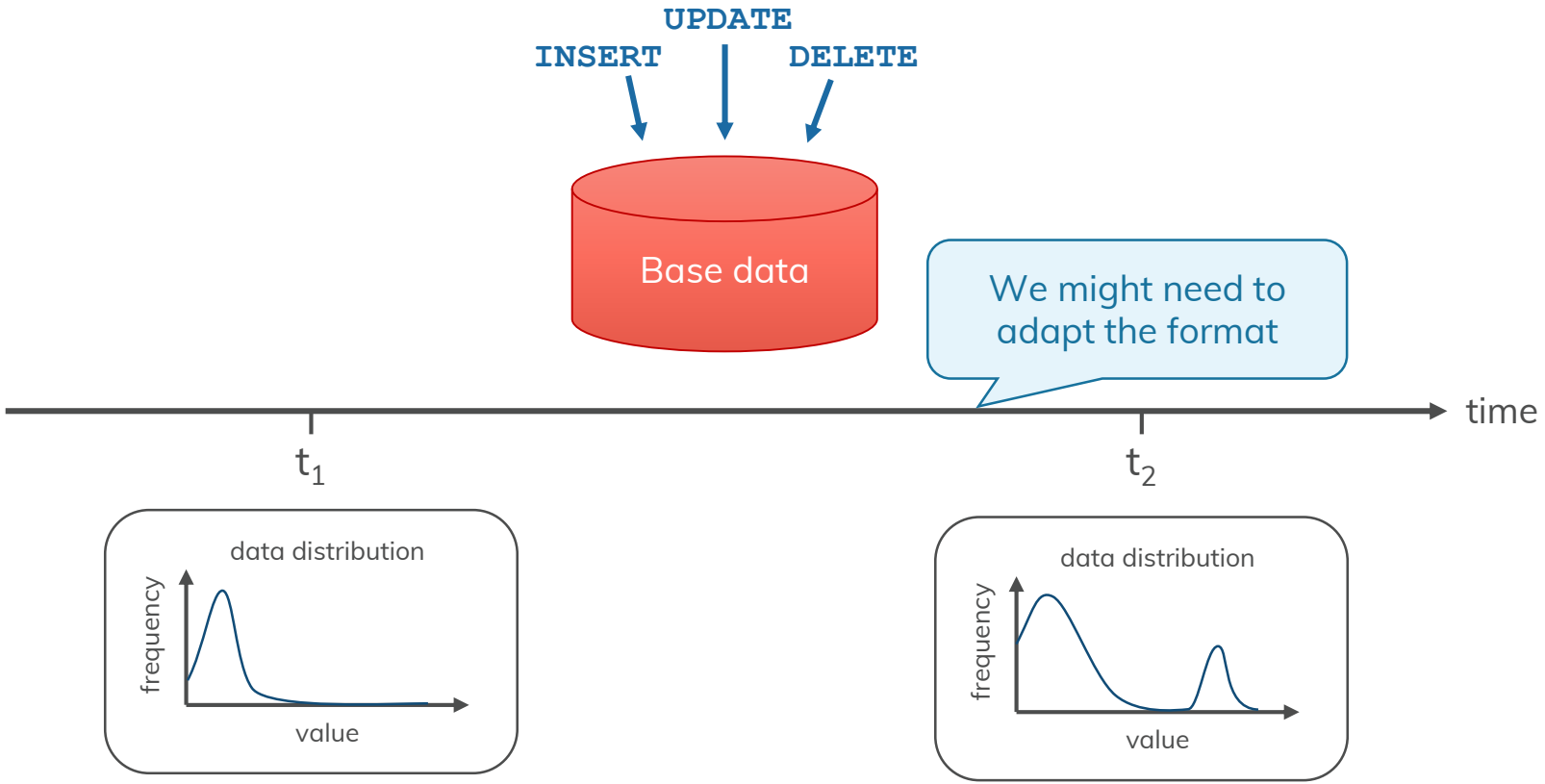
Changing Data Characteristics



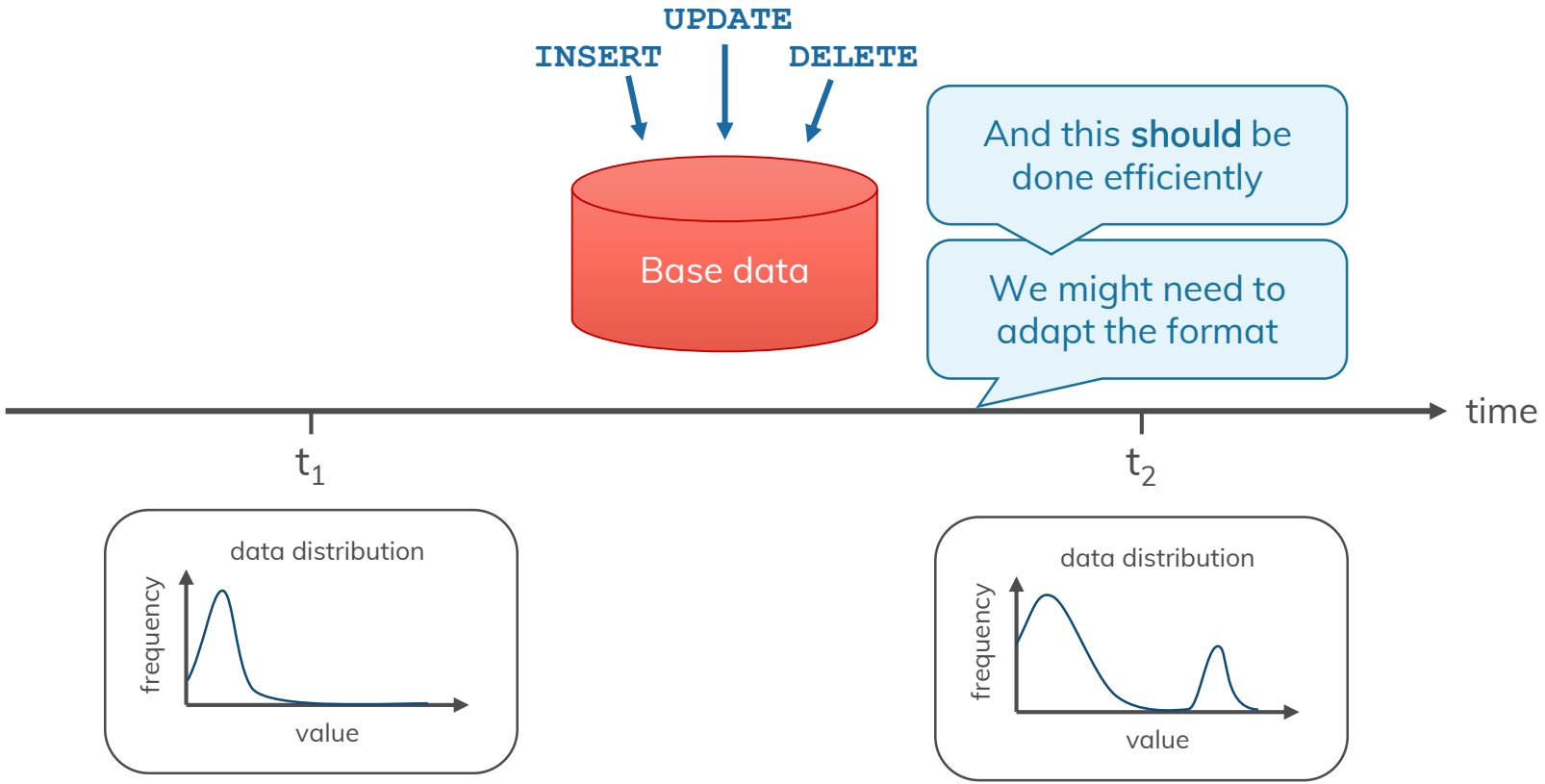
Changing Data Characteristics



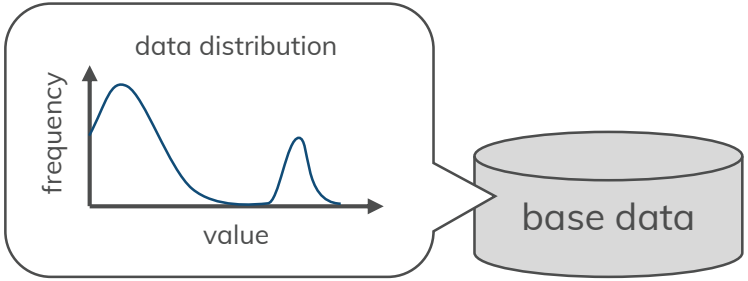
Changing Data Characteristics



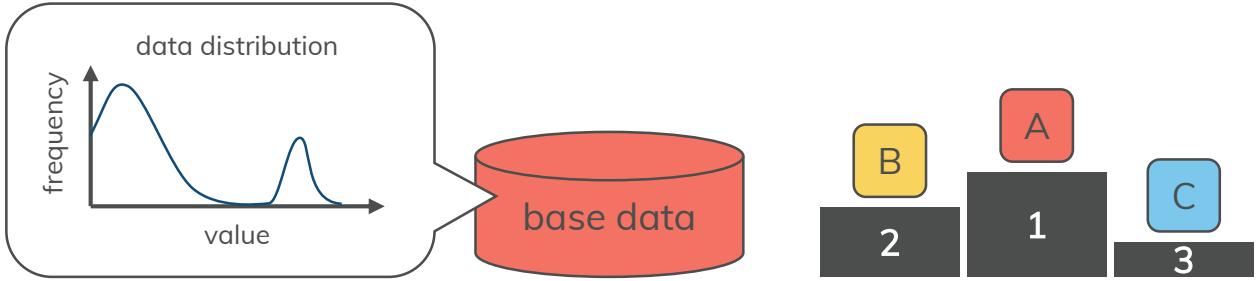
Changing Data Characteristics



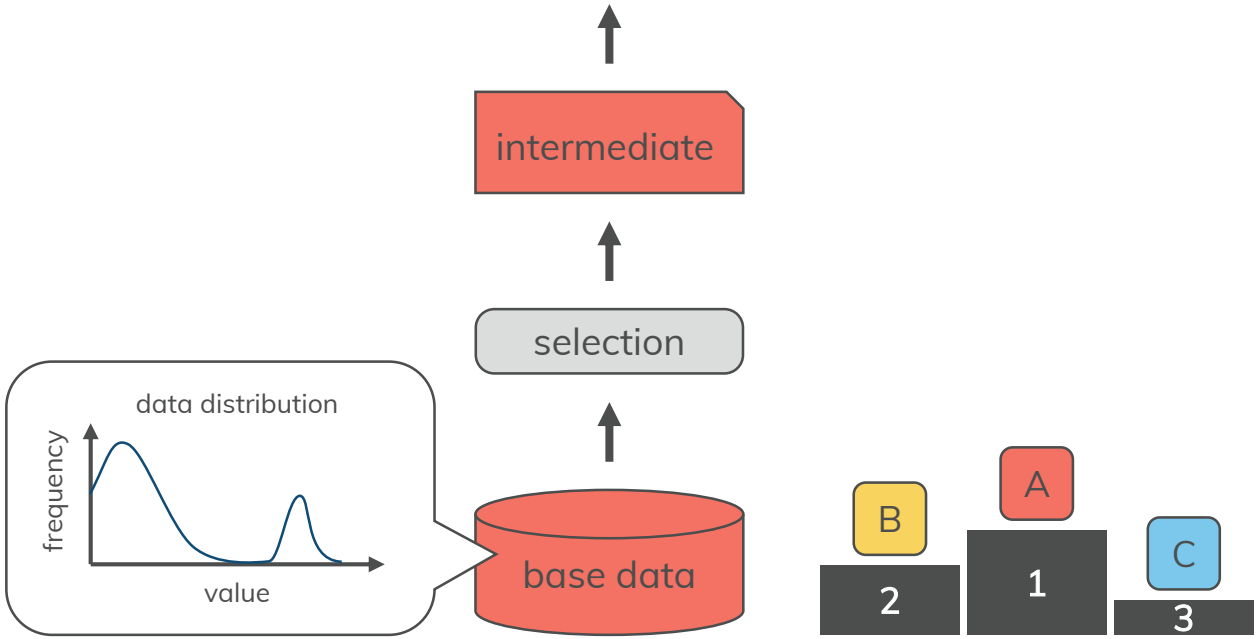
Changing Data Characteristics



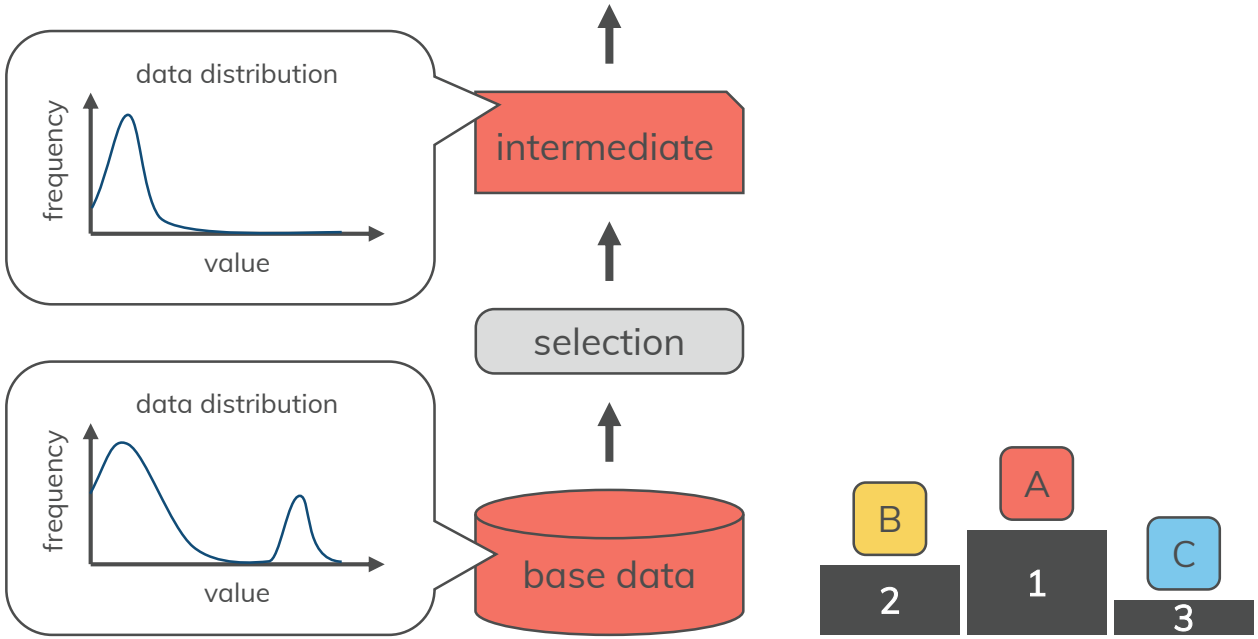
Changing Data Characteristics



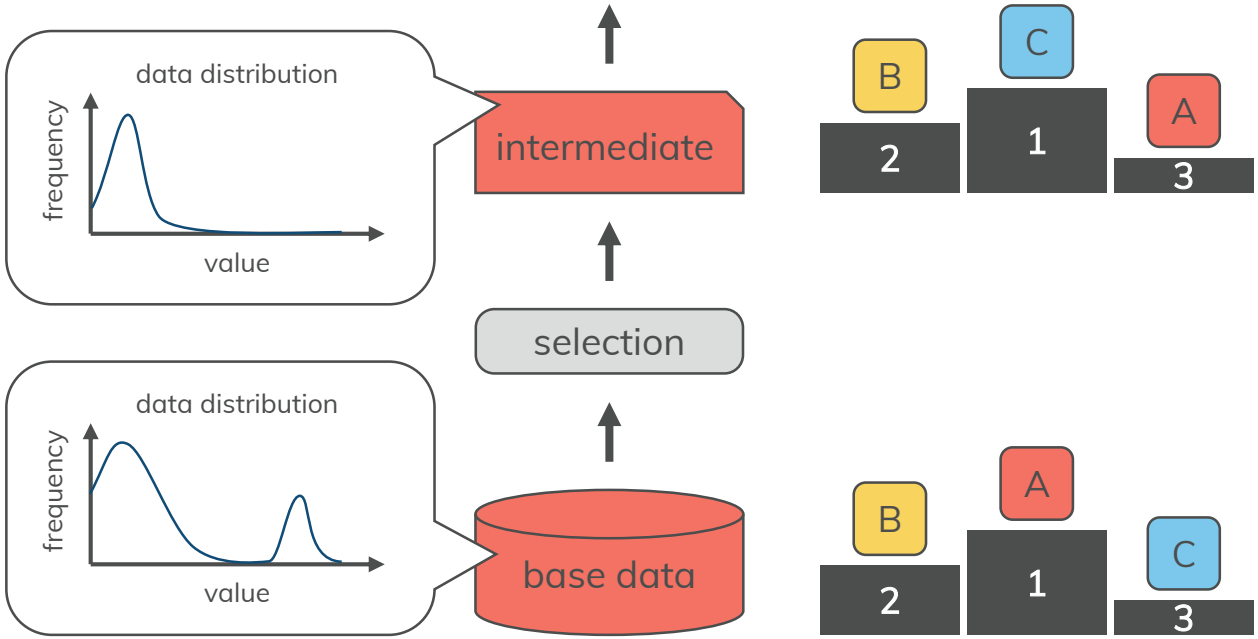
Changing Data Characteristics



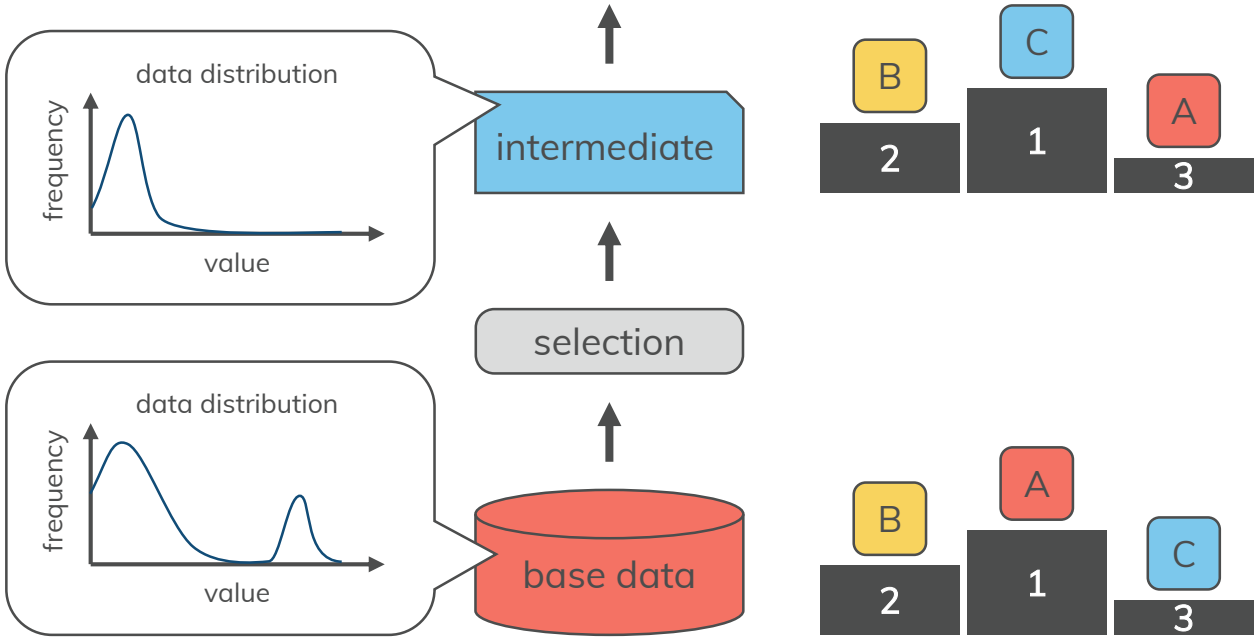
Changing Data Characteristics



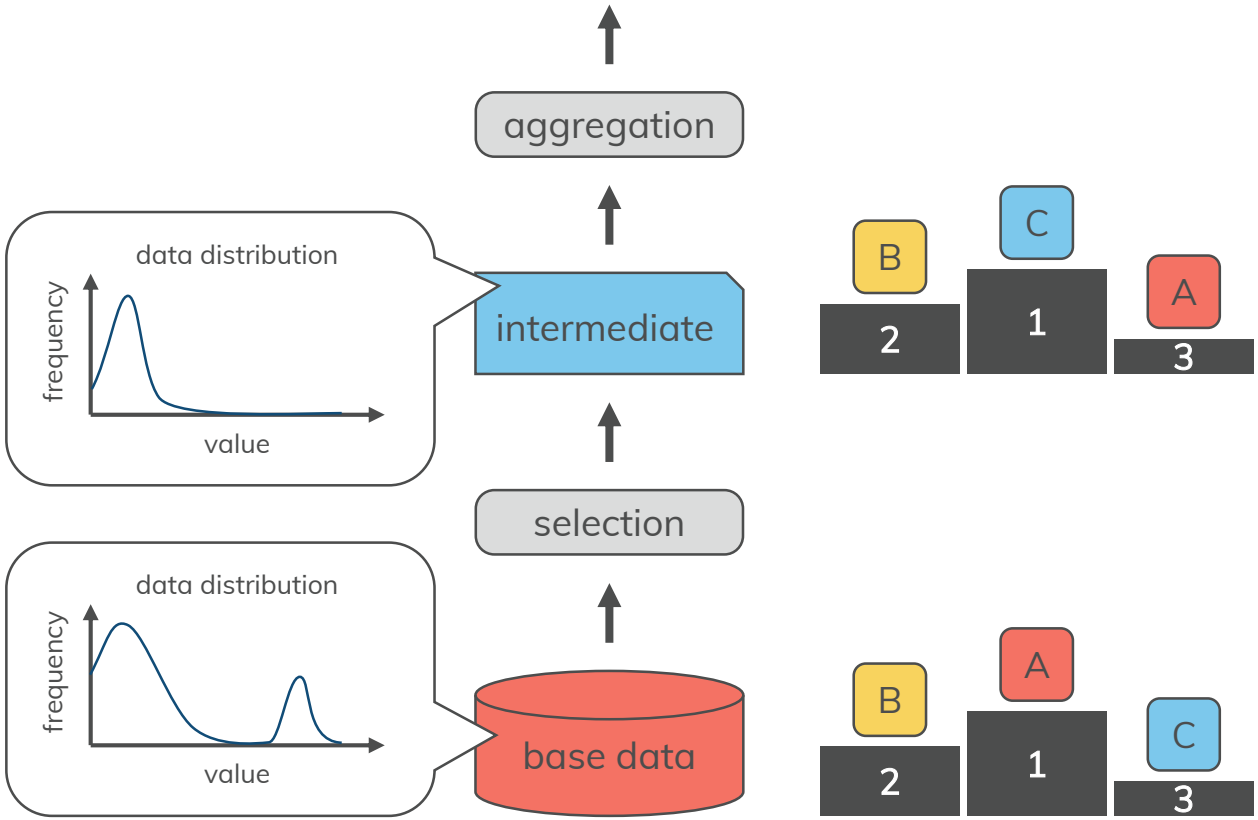
Changing Data Characteristics



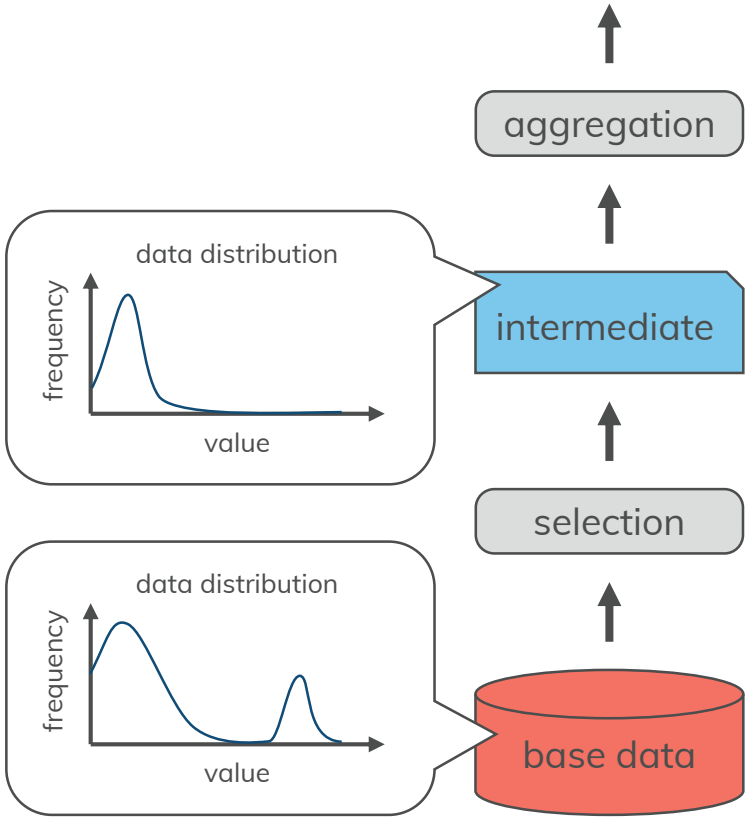
Changing Data Characteristics

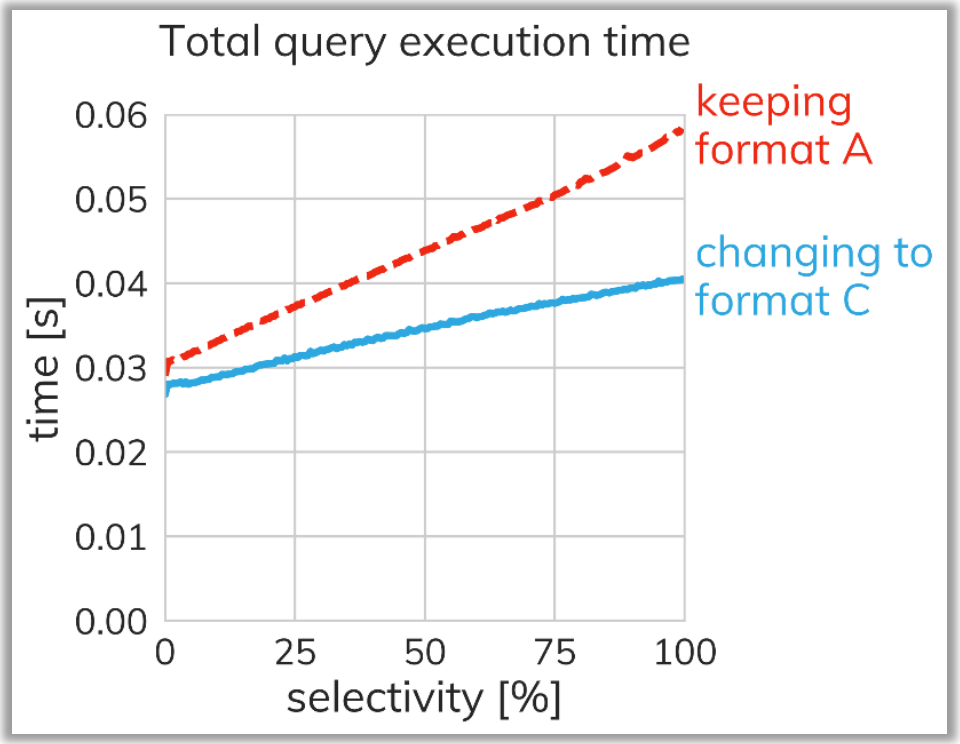


Changing Data Characteristics

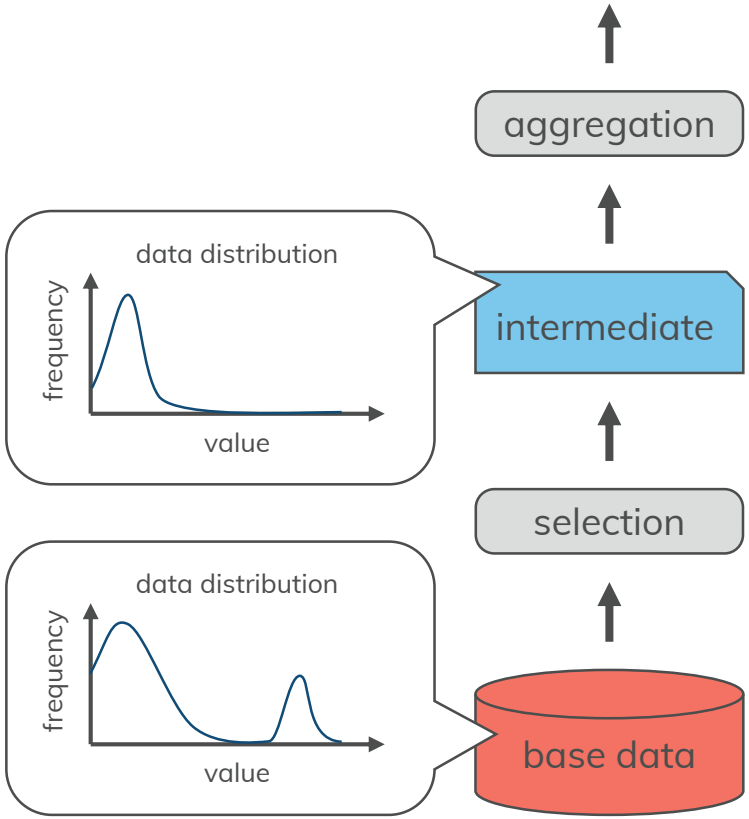


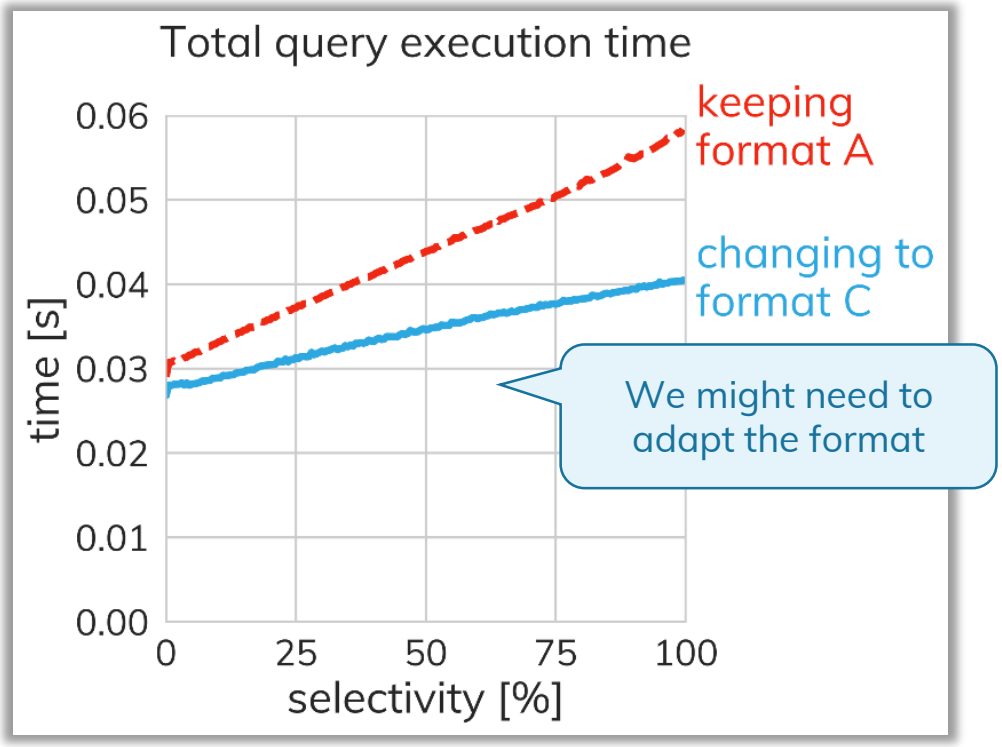
Changing Data Characteristics



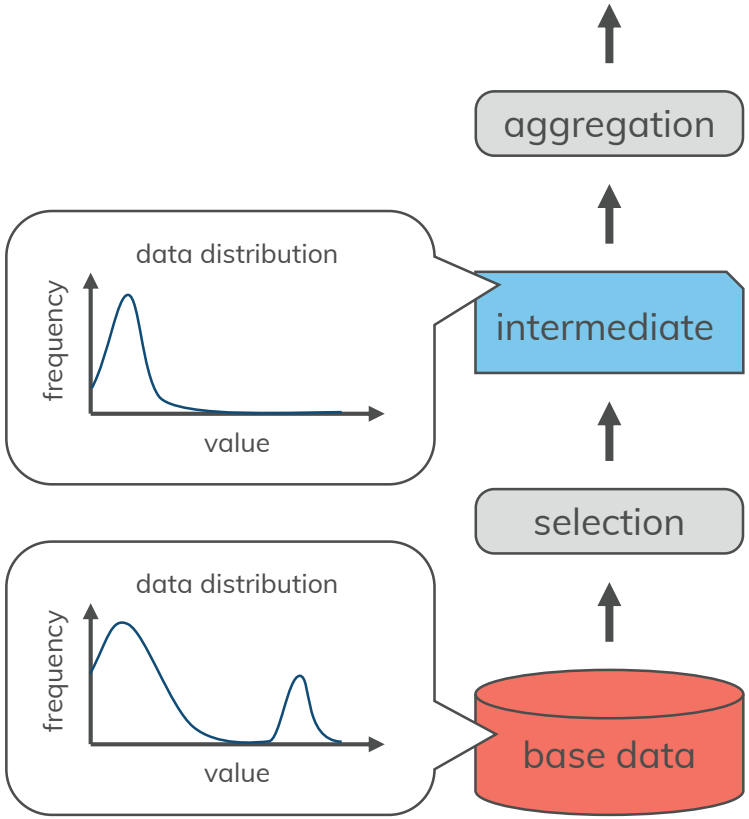


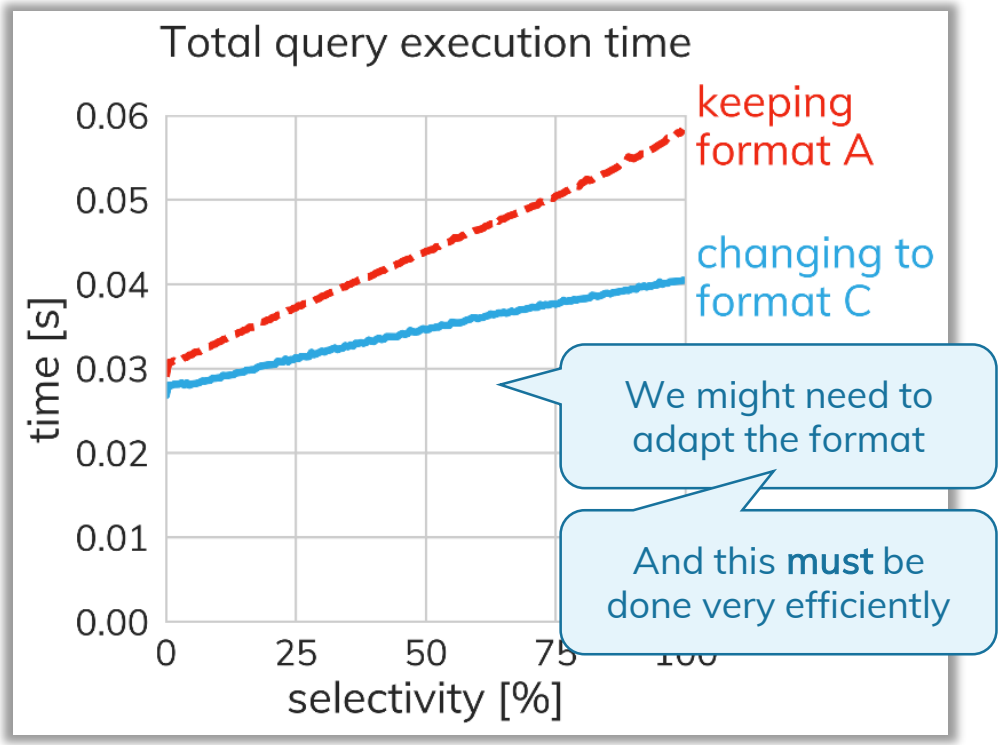
Changing Data Characteristics





Changing Data Characteristics





Data Transformation Techniques

Dresden Database Systems Group

Data Transformation

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

Data Transformation Techniques

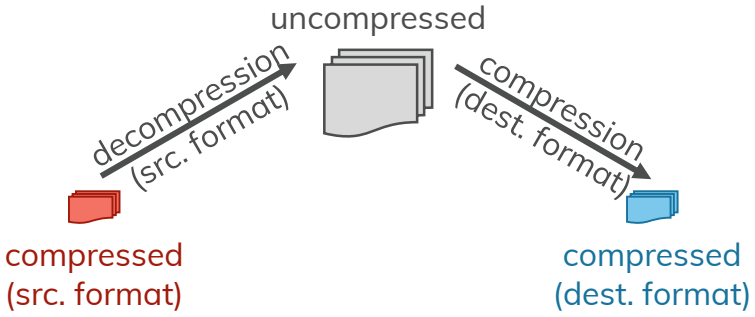
Dresden Database Systems Group

Data Transformation

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

Indirect transformation

Complete decompression and recompression



Relies only on existing (de)compression algorithms

Inefficient materialization of the uncompressed data

Data Transformation Techniques

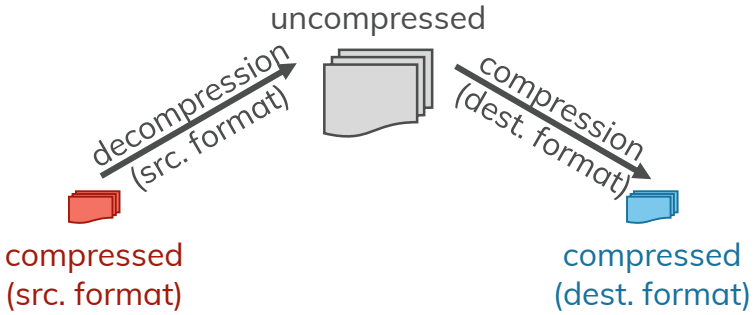
Dresden Database Systems Group

Data Transformation

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

Indirect transformation

Complete decompression and recompression



Relies only on existing (de)compression algorithms

Inefficient materialization of the uncompressed data

Direct transformation

Transformation in one step

Novel class of algorithms related to (de)compression

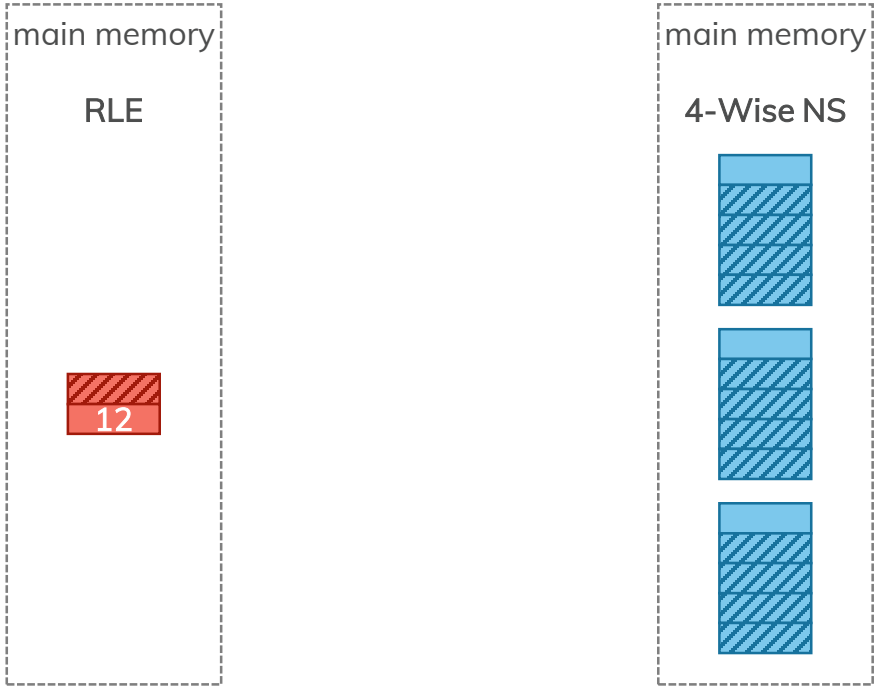
Can be much more efficient depending on the data

Example: From RLE to 4-Wise NS

Example: From RLE to 4-Wise NS

Dresden Database Systems Group

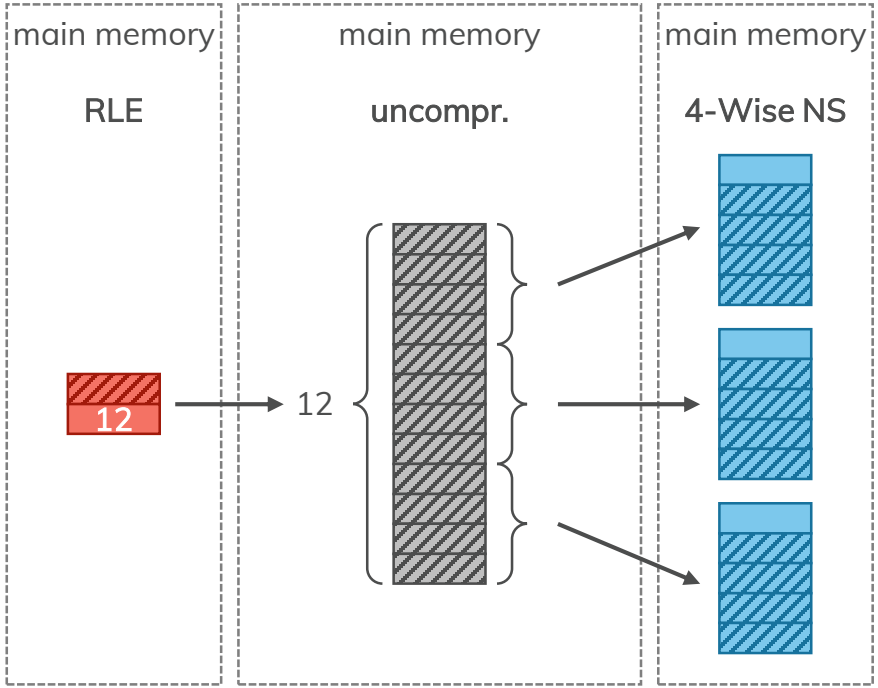
Indirect transformation



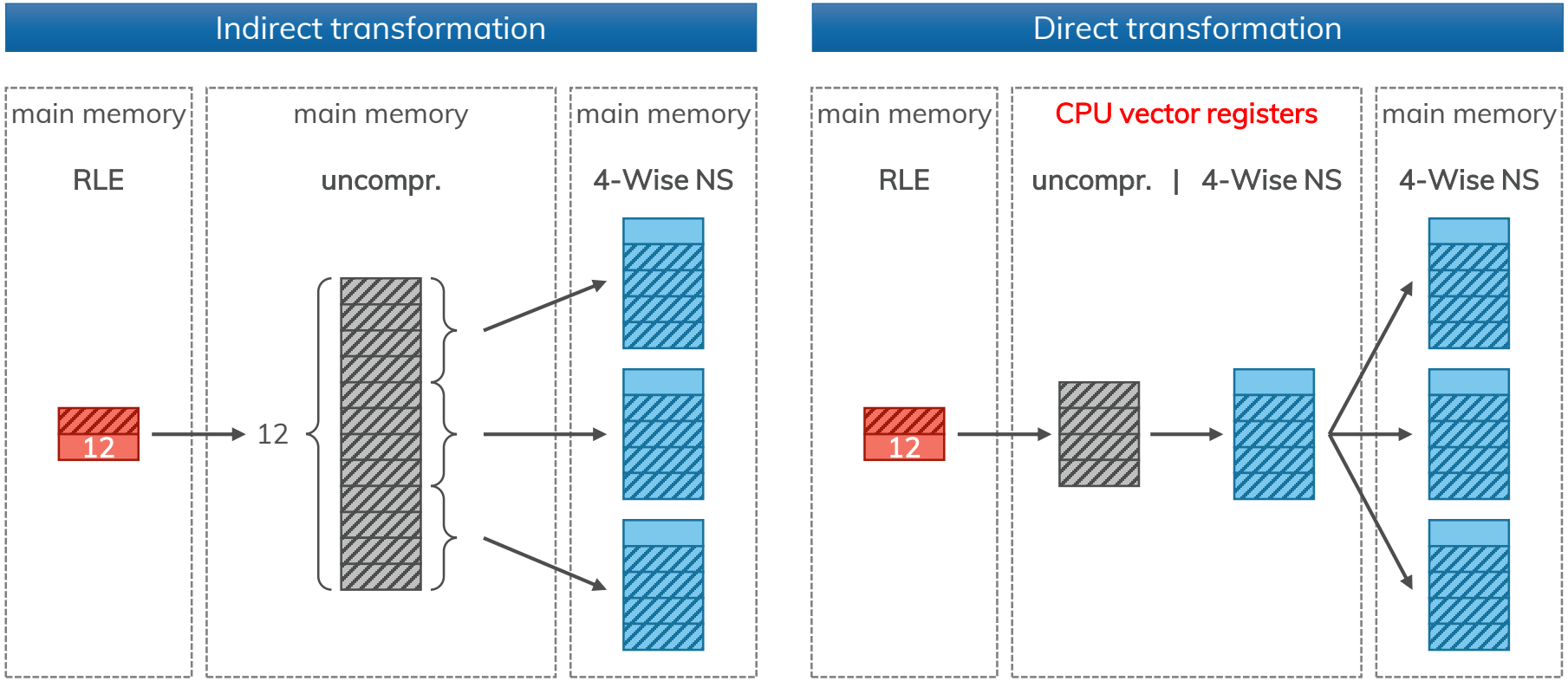
Example: From RLE to 4-Wise NS

Dresden Database Systems Group

Indirect transformation



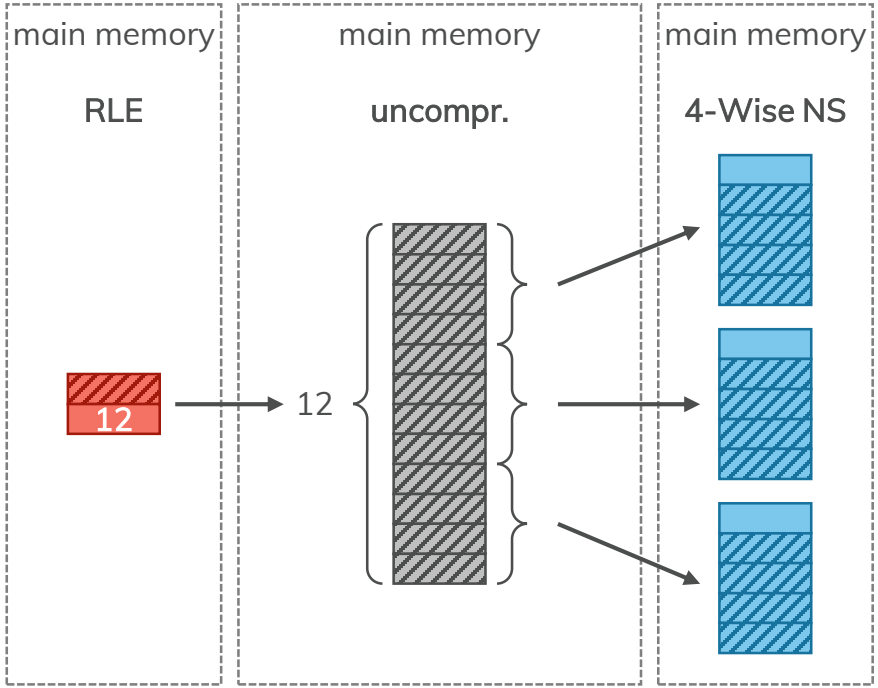
Example: From RLE to 4-Wise NS



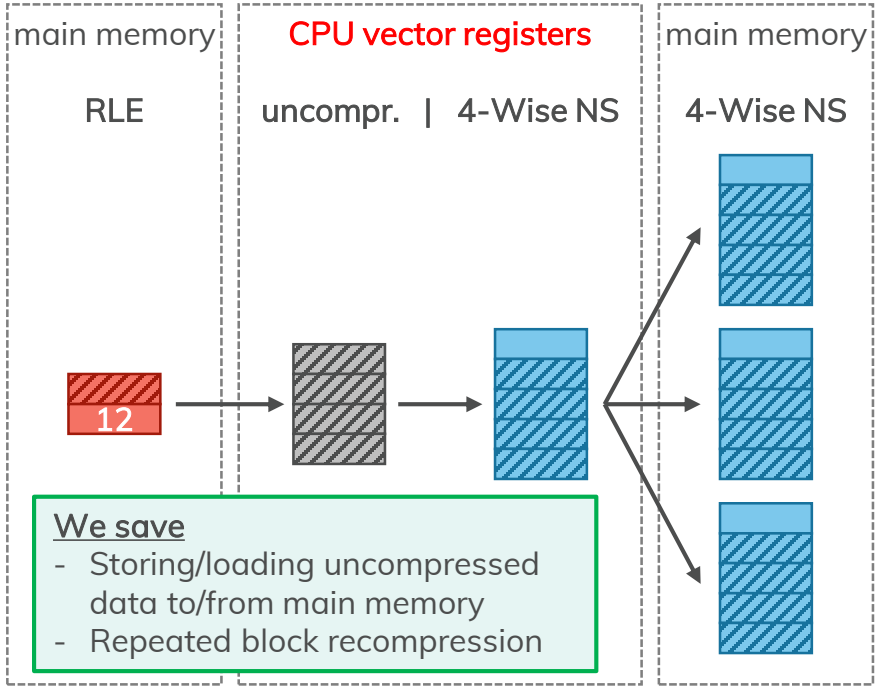
Example: From RLE to 4-Wise NS

Dresden Database Systems Group

Indirect transformation



Direct transformation



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

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization

Completed work

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

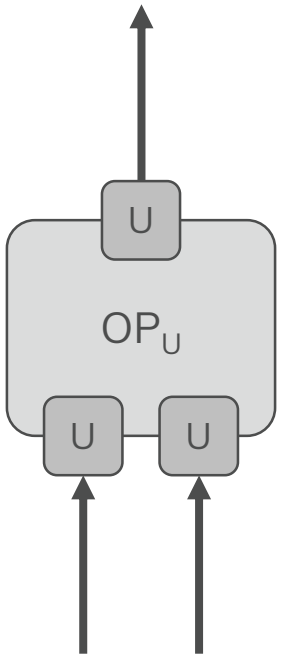
Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization
Completed work	Ongoing work	

Uncompressed Processing

Dresden Database Systems Group

Processing Model

- Column-at-a-Time
- Operators consume and produce columns



Uncompressed Processing

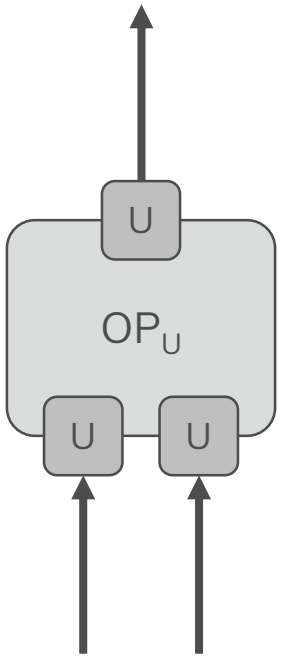
Dresden Database Systems Group

Processing Model

- Column-at-a-Time
- Operators consume and produce columns

Inputs and outputs

- Each might be represented in its individual compressed format



Uncompressed Processing

Dresden Database Systems Group

Processing Model

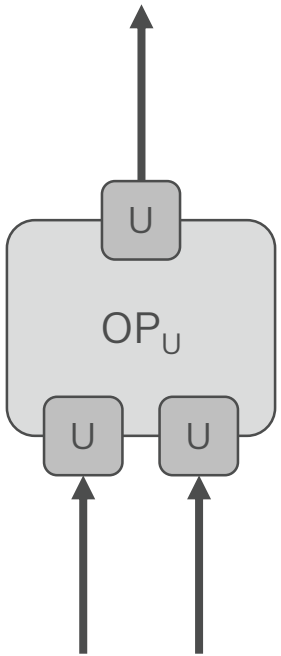
- Column-at-a-Time
- Operators consume and produce columns

Inputs and outputs

- Each might be represented in its individual compressed format

We assume

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



Uncompressed Processing

Dresden Database Systems Group

Processing Model

- Column-at-a-Time
- Operators consume and produce columns

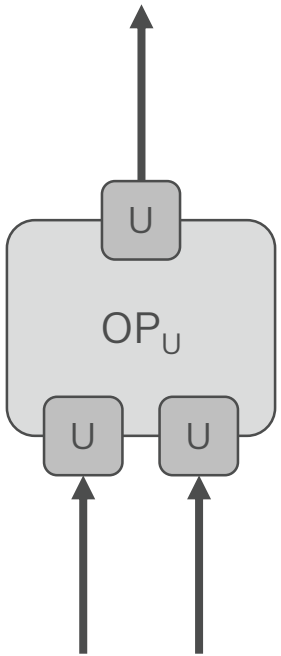
Inputs and outputs

- 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

Dresden Database Systems Group

Idea

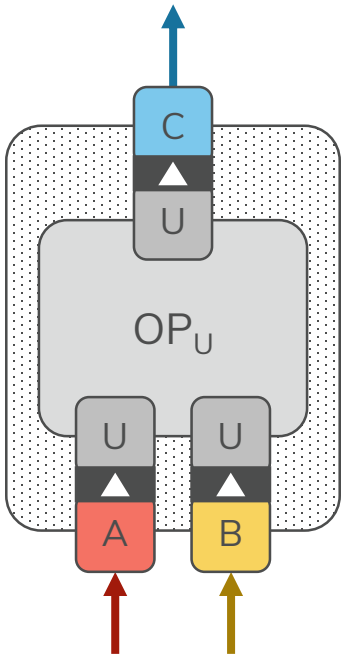
- Wrapper around original operator
  - Decompresses inputs
  - Recompresses outputs

  } temporarily
- Efficiency: work on blocks fitting into the L1-cache
- Format changes through wrapper configuration

Requires

- 1 operator variant
- n compressions
- n decompressions

} exist already



Advantage

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

Disadvantage

- Wastes potential of working directly on compressed data

Wrapper Employing Direct Transformation

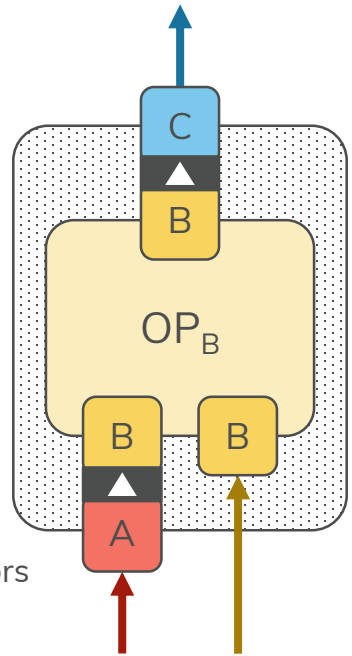
Dresden Database Systems Group

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 } reusable for other operators



Advantage

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

Disadvantage

- Wrapper still causes overhead

Specialized Operator

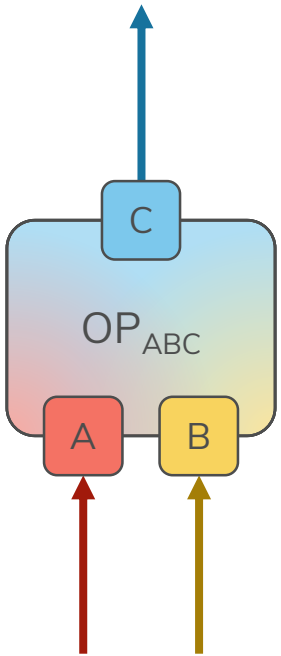
Dresden Database Systems Group

Idea

- Operator tailored to a specific format combination
- No wrapper

Requires

- n^{i+o} operator variants



Advantage

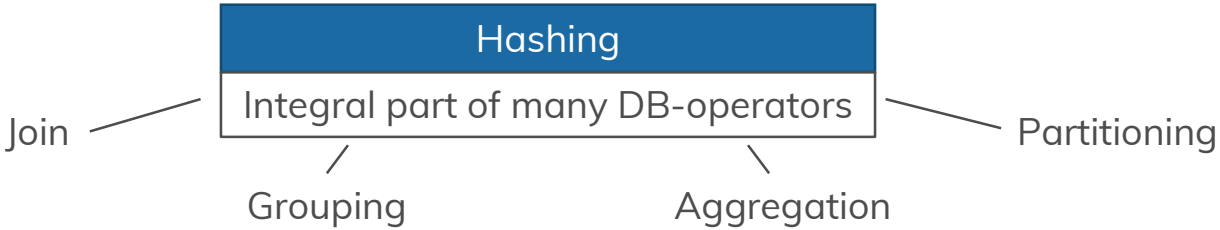
- Maximum efficiency

Disadvantage

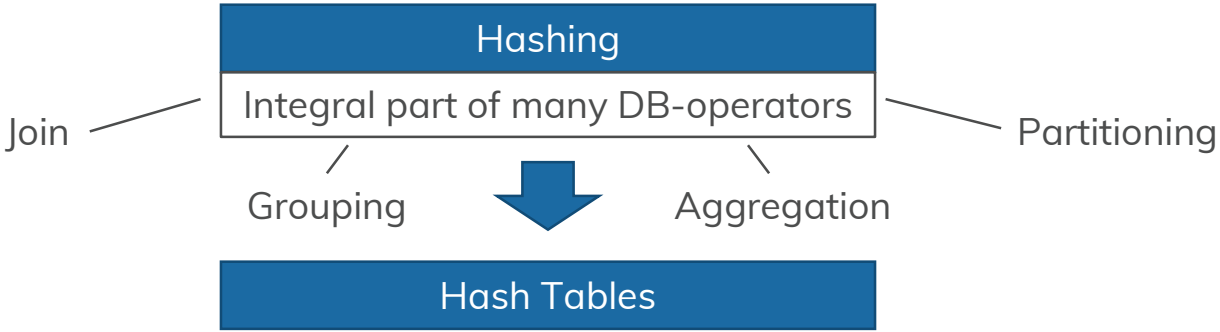
- Highest integration effort due to high number of operator variants

Next Steps: …

Next Steps: Hashing and Compression



Next Steps: Hashing and Compression



Next Steps: Hashing and Compression

Hashing

Integral part of many DB-operators

Join

Grouping

Aggregation

Partitioning

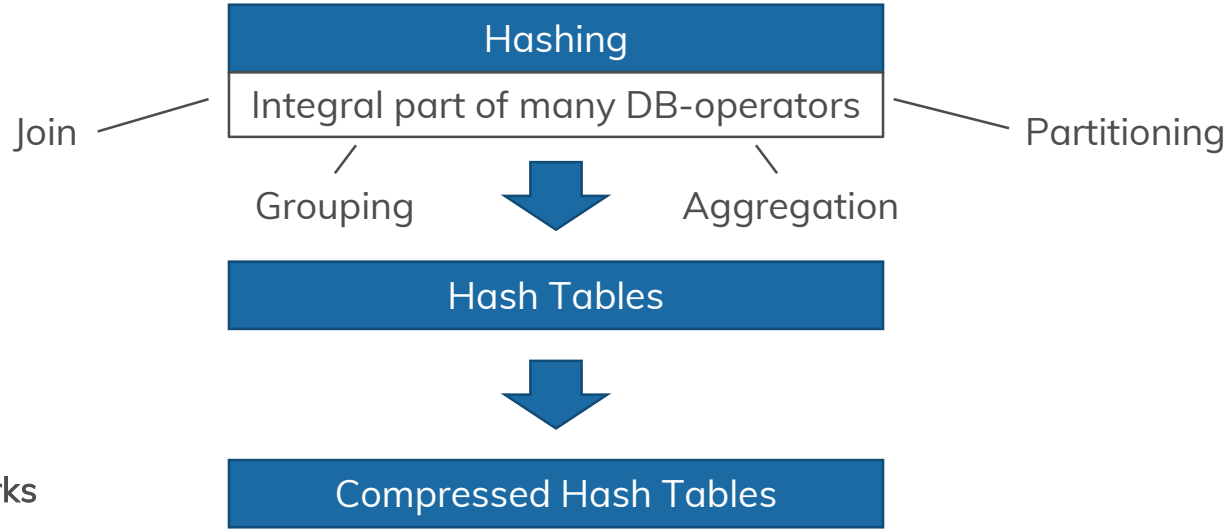
Hash Tables

Compressed Hash Tables

- To fit hash table closer to CPU in memory hierarchy
- Important due to random access pattern

Next Steps: Hashing and Compression

Dresden Database Systems Group



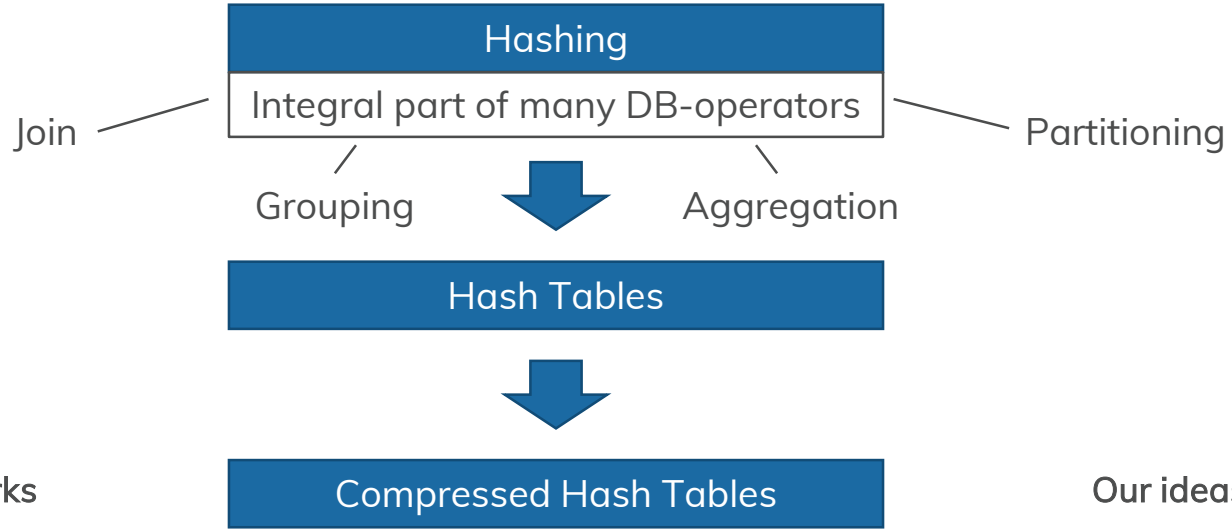
Existing works

- Concise Hash Table by IBM
- Eliminates empty buckets in huge virtual linear probing hash table
- No compression of the entries

- To fit hash table closer to CPU in memory hierarchy
- Important due to random access pattern

Next Steps: Hashing and Compression

Dresden Database Systems Group



Existing works

- Concise Hash Table by IBM
- Eliminates empty buckets in huge virtual linear probing hash table
- No compression of the entries

- To fit hash table closer to CPU in memory hierarchy
- Important due to random access pattern

Our ideas

- Compressed entries
- Splitting hash table into several ones: one per key bit width
- Storing compressed hash table entirely in processor (vector) registers, possible?

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization
Completed work	Ongoing work	

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

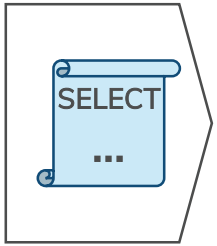
Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization
Completed work	Ongoing work	Completed/Future work

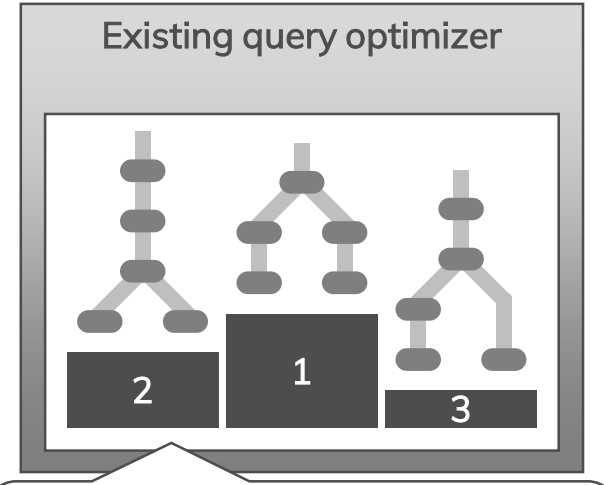
Compression-Aware Query Optimization

Compression-Aware Query Optimization

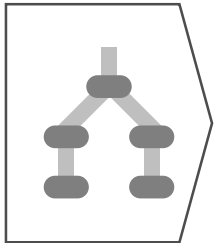
Compression-Aware Query Optimization



SQL query

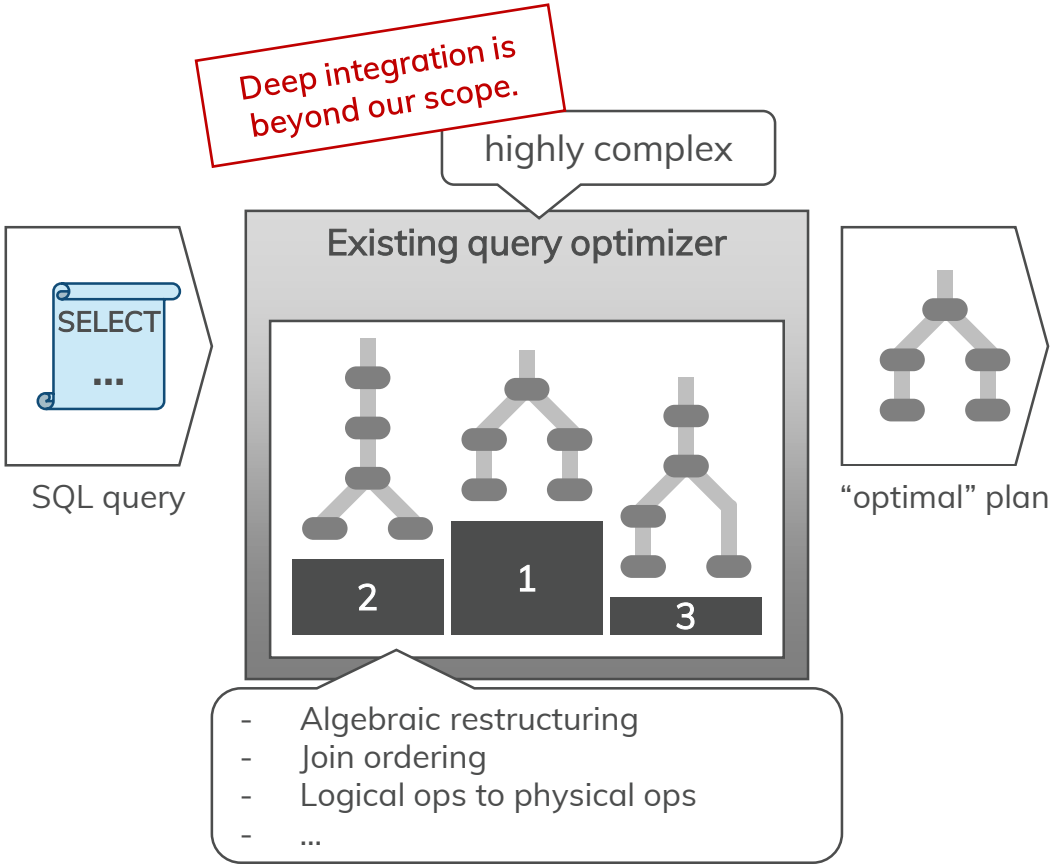


- Algebraic restructuring
- Join ordering
- Logical ops to physical ops
- ...



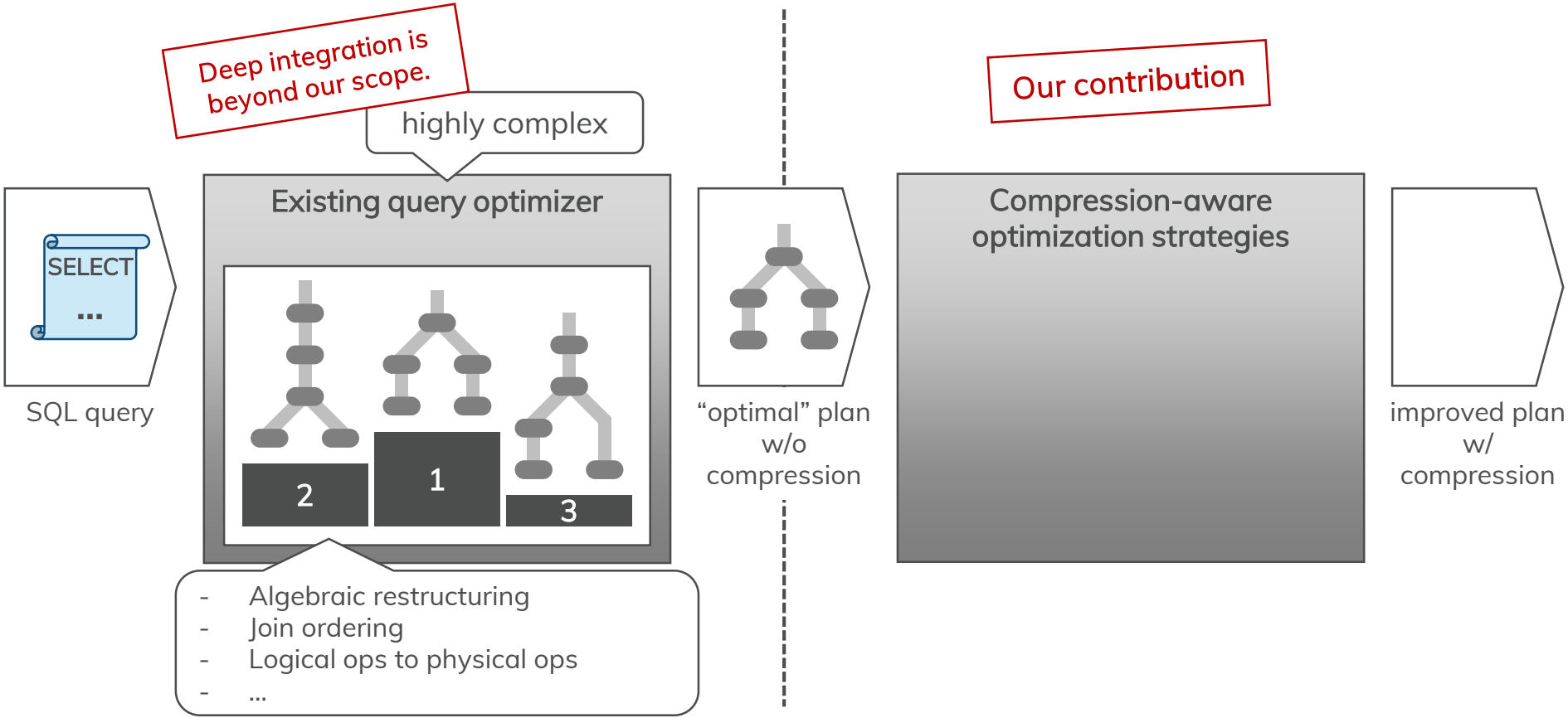
“optimal” plan

Compression-Aware Query Optimization



Compression-Aware Query Optimization

Dresden Database Systems Group

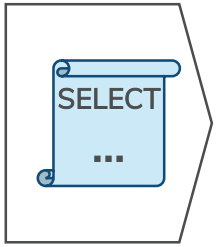


Compression-Aware Query Optimization

Dresden Database Systems Group

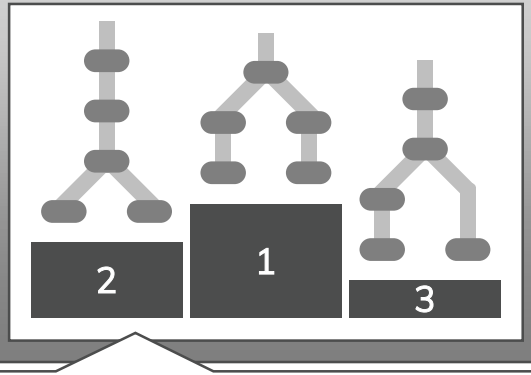
Deep integration is beyond our scope.

highly complex

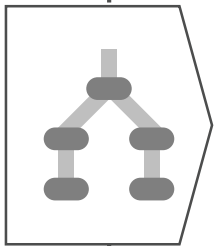


SQL query

Existing query optimizer



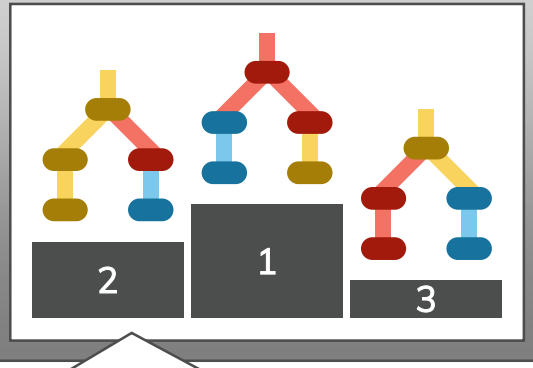
- Algebraic restructuring
- Join ordering
- Logical ops to physical ops
- ...



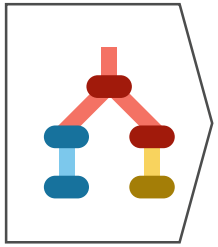
"optimal" plan w/o compression

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

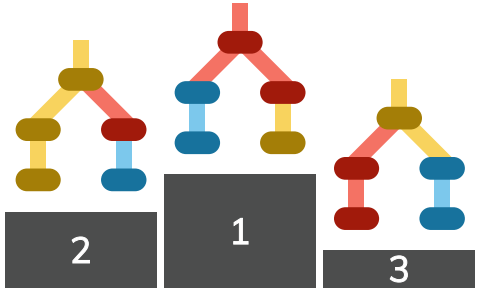
Dresden Database Systems Group

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

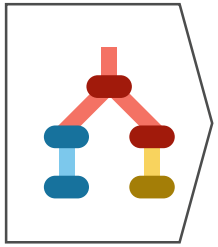
System side
- Current system load
- Optimize run time with limited memory consumption
- Optimize memory consumption with limited run time

Our contribution

Compression-aware optimization strategies



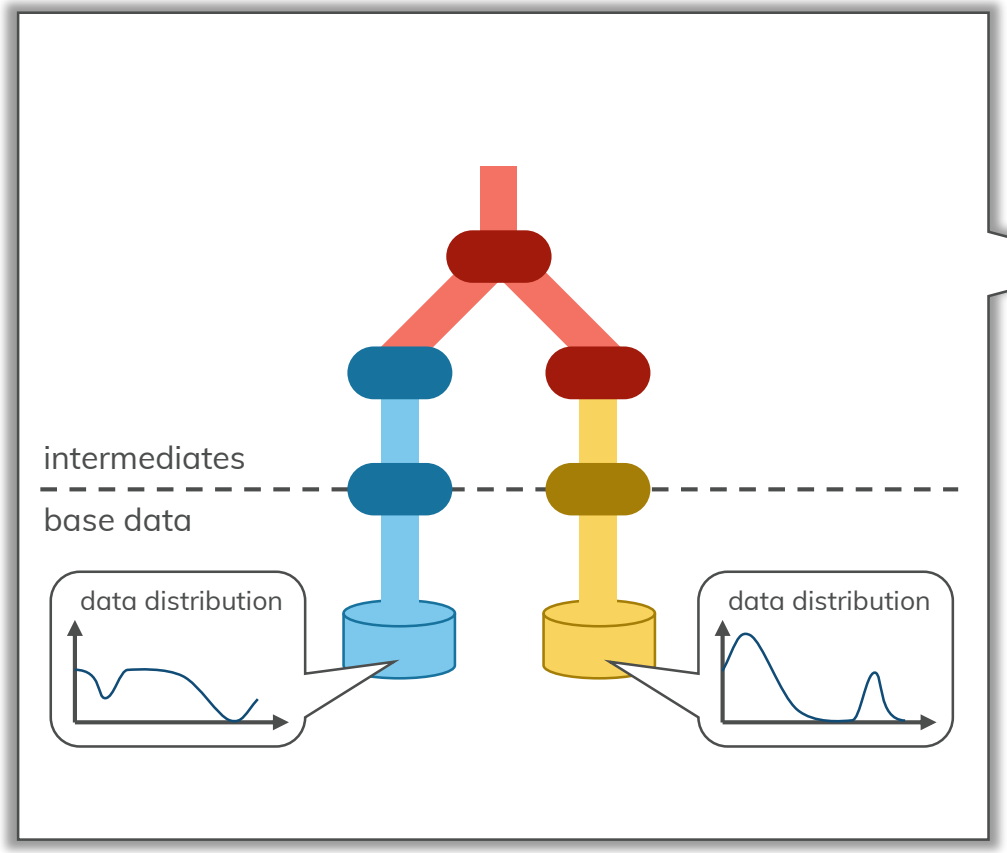
- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data



improved plan w/ compression

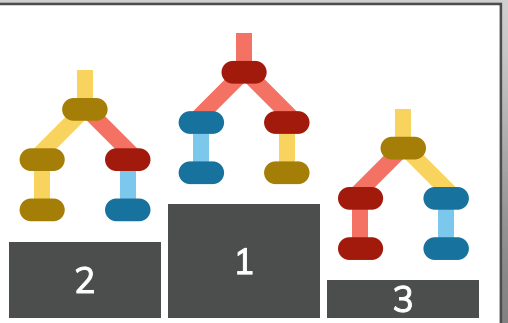
Challenges to Be Addressed

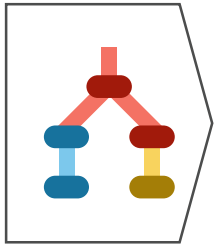
Dresden Database Systems Group



Our contribution

Compression-aware optimization strategies





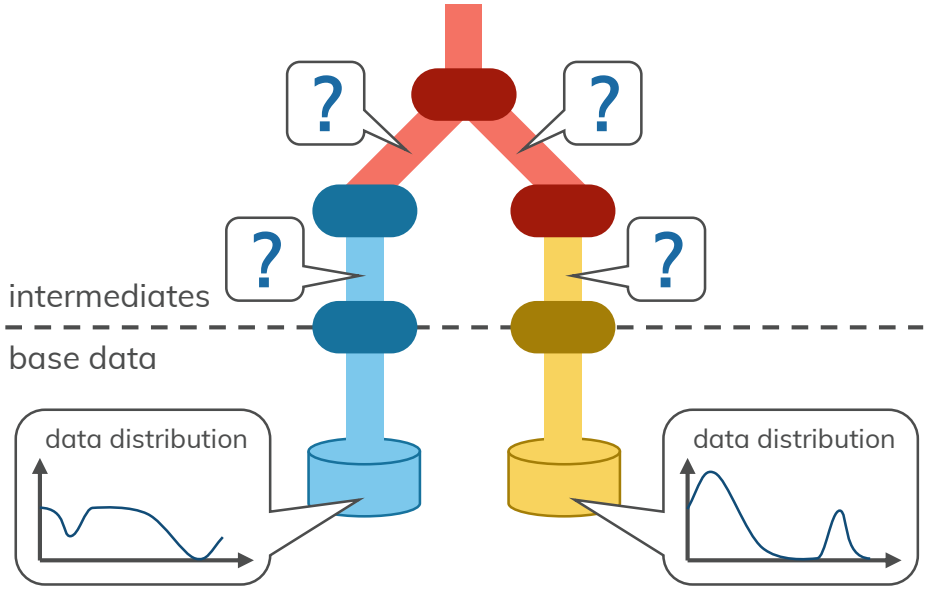
improved plan w/ compression

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

Challenges to Be Addressed

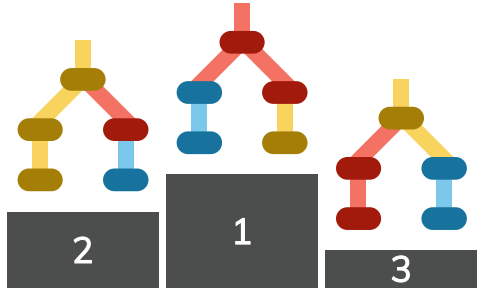
Dresden Database Systems Group

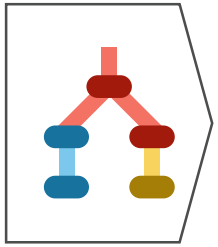
Estimation of the intermediates' data characteristics



Our contribution

Compression-aware optimization strategies



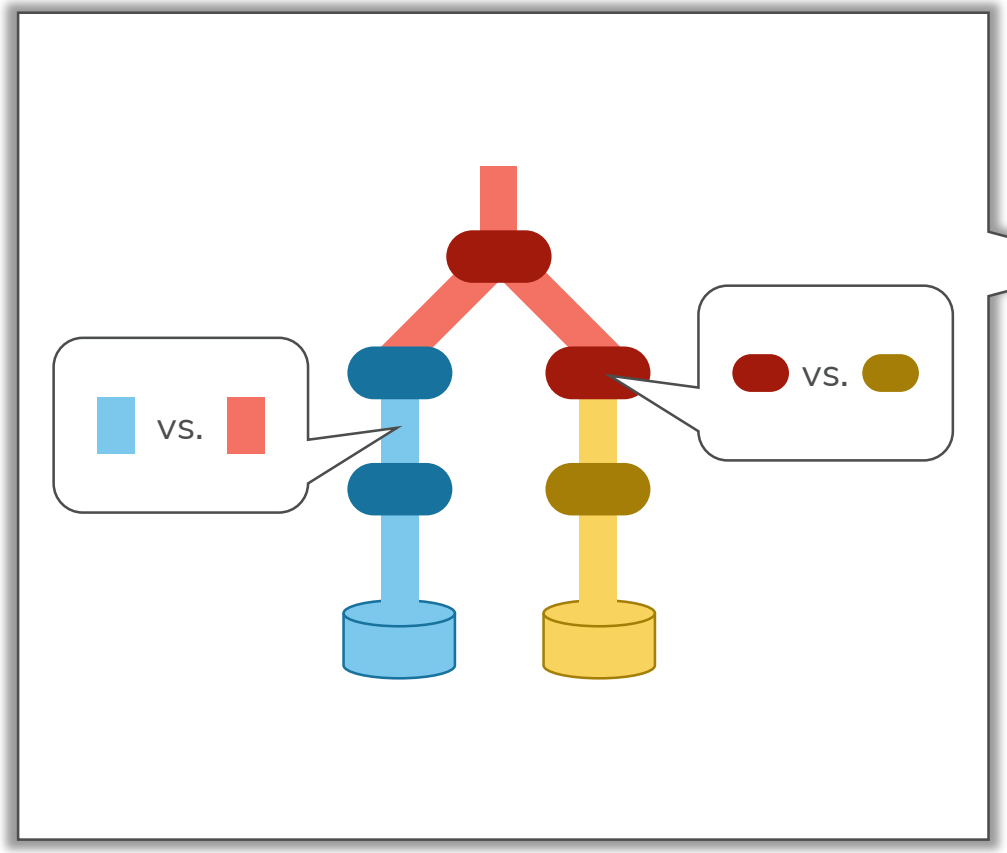


improved plan w/ compression

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

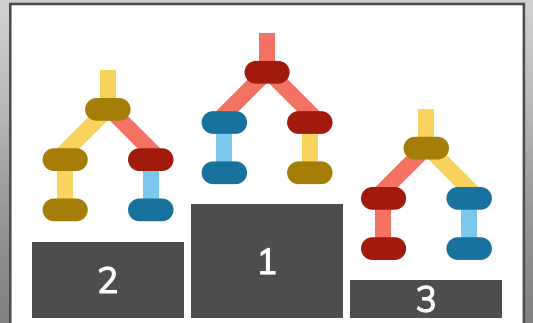
Challenges to Be Addressed

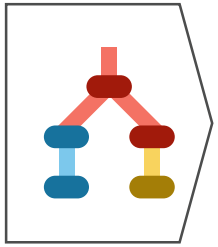
Dresden Database Systems Group



Our contribution

Compression-aware optimization strategies





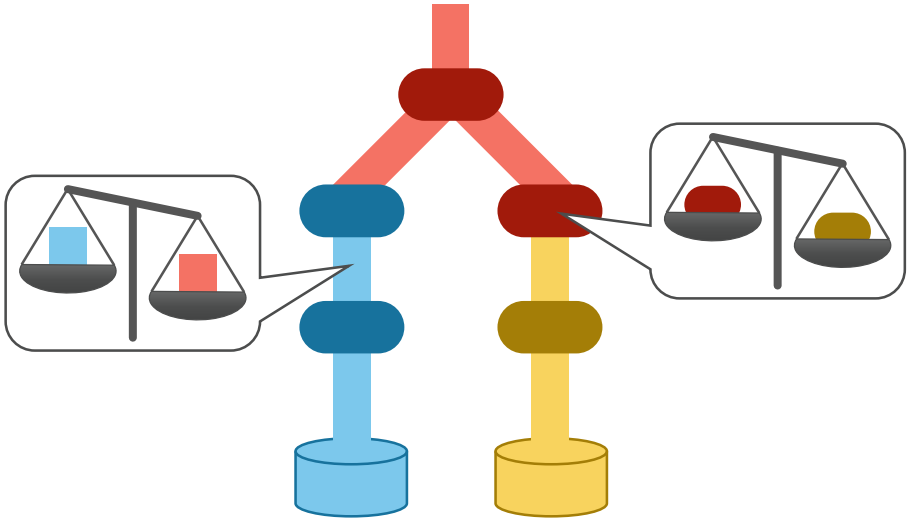
improved plan w/ compression

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

Challenges to Be Addressed

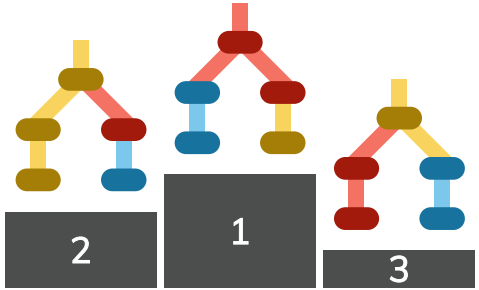
Dresden Database Systems Group

Cost model for compression algorithms and operators

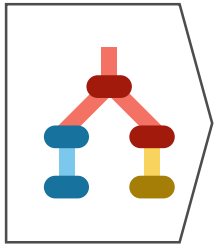


Our contribution

Compression-aware optimization strategies



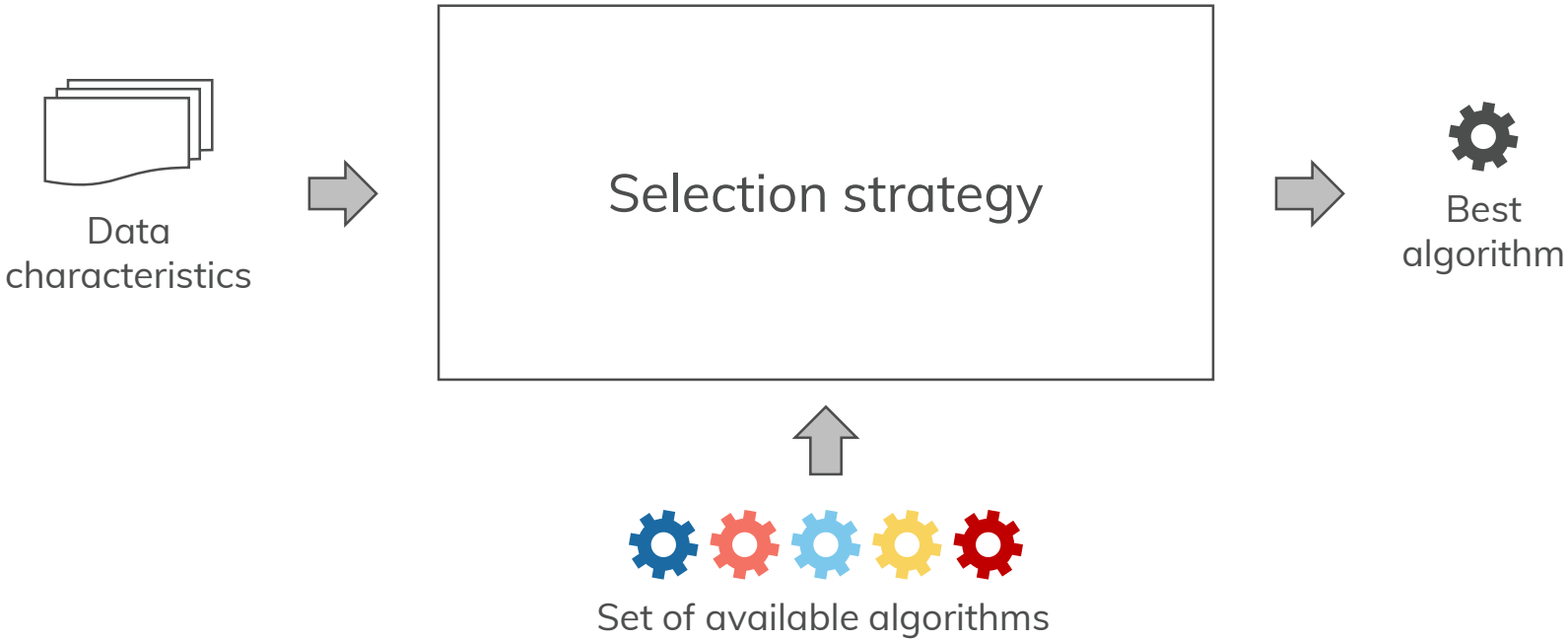
- Select suitable compressed format for each intermediate
- Replace physical operators by our operators for compressed data



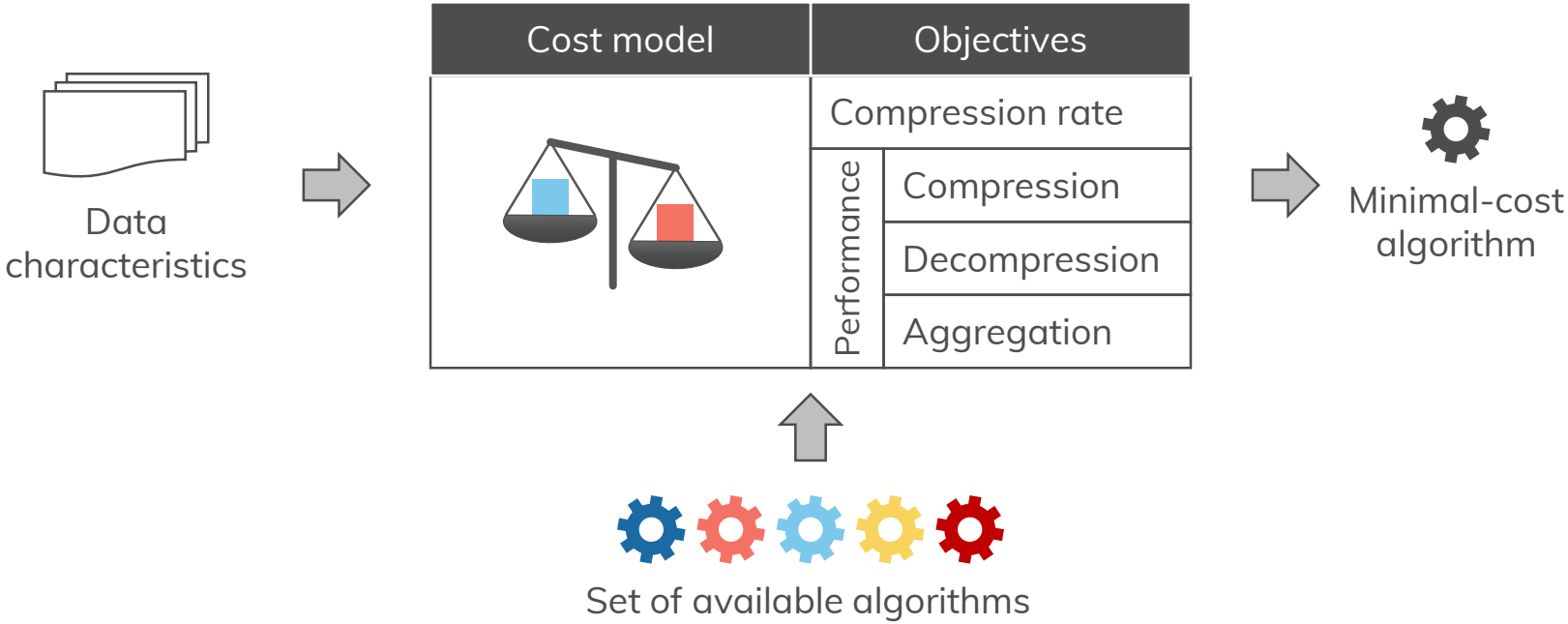
improved plan w/ compression

Selecting a Suitable Compression Algorithm

Selecting a Suitable Compression Algorithm



Selecting a Suitable Compression Algorithm



Grey-Box Approach

Grey-Box Approach

Dresden Database Systems Group

Modelling

Everything known from the algorithm's specification
(and our experimental survey)

Grey-Box Approach

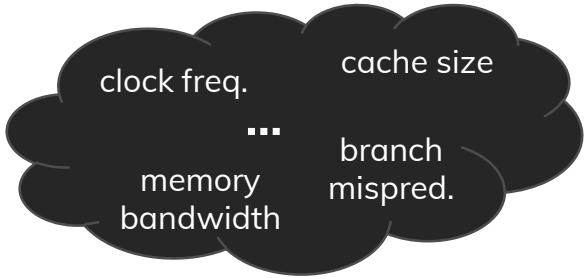
Dresden Database Systems Group

Modelling

Everything known from the algorithm's specification
(and our experimental survey)

Measuring

Everything related to the algorithm's execution on the hardware



One-time calibration phase

Selecting a Suitable Compression Algorithm

Algorithms' profiles (calibration)

Data characteristics

Cost model	Objectives	
	Compression rate	
	Performance	Compression
		Decompression
		Aggregation

Minimal-cost algorithm

Set of available algorithms

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization
Completed work	Ongoing work	Completed/Future work

Our Vision

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Requires a Context-sensitive DBMS

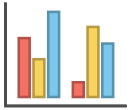
Data characteristics	Surrounding operators	System utilization, user preferences
Structural Aspect	**Operational Aspect**	**Optimization Aspect**
Investigation of lightweight compression	Integration of compression into query execution	Compression-aware query optimization
Completed work	Ongoing work	Completed/Future work

Summary and Contributions

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Contributions so far



Experimental survey
of lightweight compr. algos

EDBT'17
(ACM TODS)

Direct transformations
for compressed data

ADBIS'15

Cost model
for compression algorithms

(ACM TODS)

Benchmark framework
for lightweight compr. algos

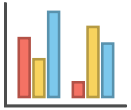
TPCTC'15

Summary and Contributions

Dresden Database Systems Group

Our Vision: Balanced Query Processing based on Compressed Intermediate Results

Contributions so far



Experimental survey
of lightweight compr. algos

EDBT'17
(ACM TODS)

Direct transformations
for compressed data

ADBIS'15

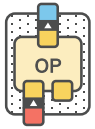
Cost model
for compression algorithms

(ACM TODS)

Benchmark framework
for lightweight compr. algos

TPCTC'15

Contributions still in the works



Operators for compressed data
with different degrees of integration

Cost model
for compression-aware operators

Compression-aware strategies
for the query optimizer