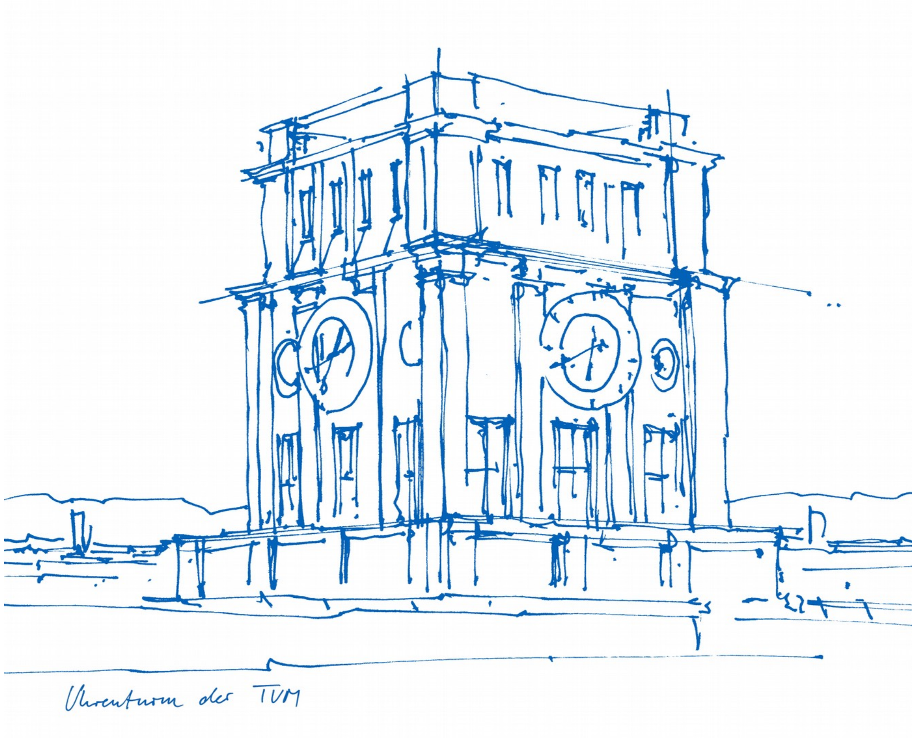
Data Blocks: Hybrid OLTP and OLAP on compressed storage

Ben Brümmer

Technische Universität München

Fürstenfeldbruck, 26. November 2018



Problem

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

Solution

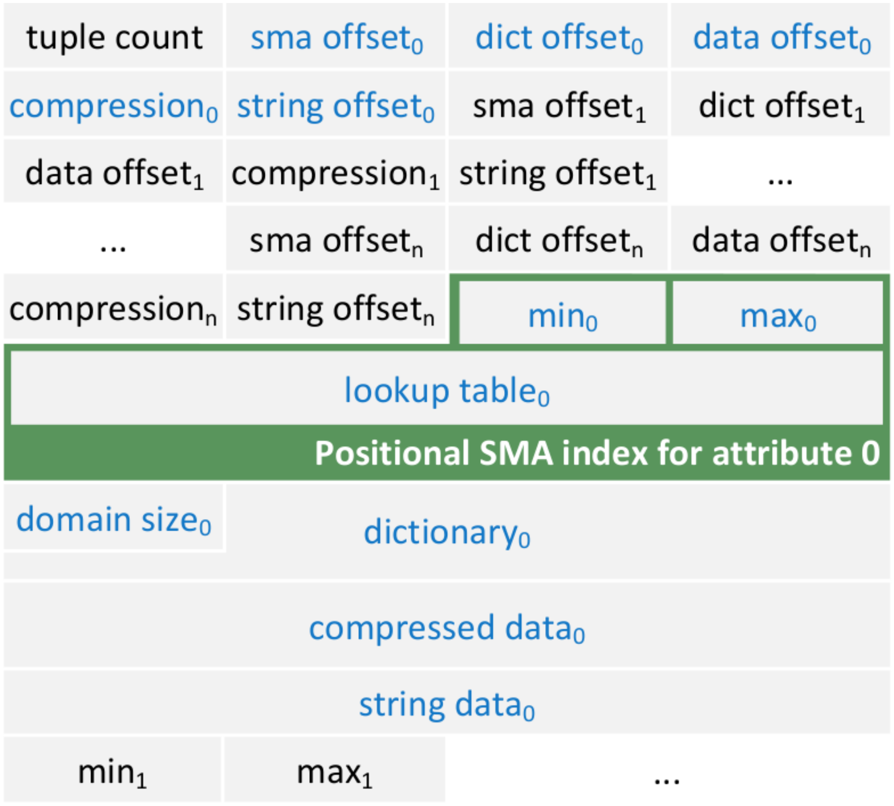
Design a storage format for cold data with:

- Compression (fit into main memory)
- High point accessibility, mainly for OLTP but also OLAP requests (fast access)
-

=> Data Blocks

How are Data Blocks constructed?

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



[1] Paper: Figure 3

Compression

Different method chosen based on value distribution (per attribute)

Three different compression methods:

- Single value compression
- Ordered dictionary compression
- Truncation

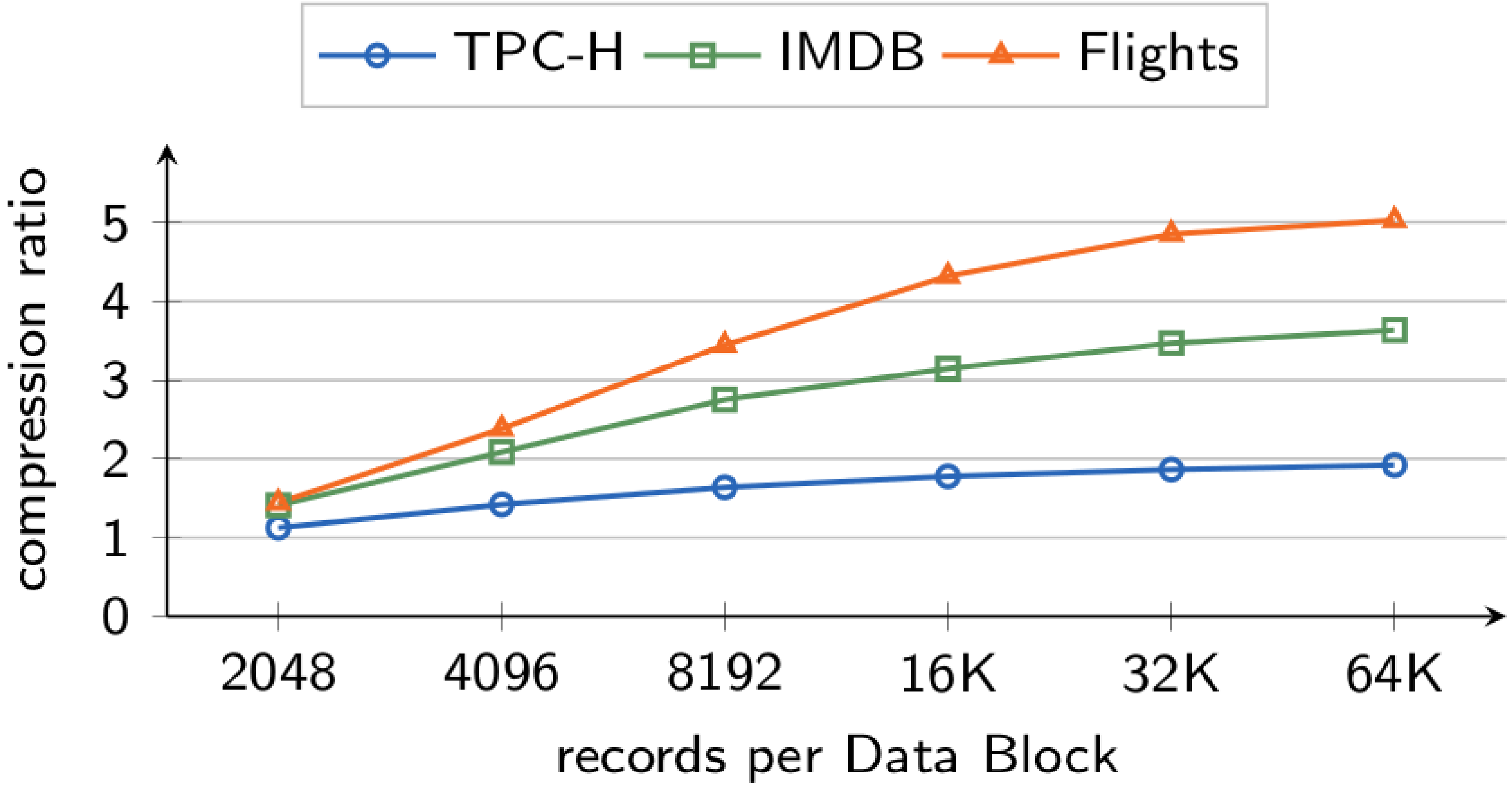
➔ Possibility of point access (no sub-byte encoding)

➔ SIMD operations are usable on compressed data

Compression

Data Base	data	Uncompressed	Compressed
Vectorwise	TPC-H 100	105 GB	66 GB
	IMDB Cast	0.72 GB	0.24 GB
	Flights	11 GB	3,2 GB
Hyper	TPC-H 100	126 GB	66 GB
	IMDB Cast	1.8 GB	0.5 GB
	Flights	21 GB	4.2 GB

Compression



[1] Paper: Figure 10

SMA and PSMA

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

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

PSMA

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

- For each byte of attribute data type 2^8 entries, e.g. 4-byte int - > 4 * 2^8 entries
- Each entry is a range

Building a lookup table

- Initialize table with 0 ranges

SMA min: 2

00	00	00	00

Delta bytes

5	20	2	4	290	8	5	420	3	10

Attribute data

		# of values mapping to entry
0	[0 , 0)	1
1	[0 , 0)	1
2	[0 , 0)	1
3	[0 , 0)	1
...		
257	[0 , 0)	2^8
...		
1024	[0 , 0)	2^{24}

Lookup table

Building a lookup table

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

SMA min: 2

00	00	00	00

Delta bytes

5	20	2	4	290	8	5	420	3	10

Attribute data

		# of values mapping to entry
0	[0 , 0)	1
1	[0 , 0)	1
2	[0 , 0)	1
3	[0 , 0)	1
...		
257	[0 , 0)	2^8
...		
1024	[0 , 0)	2^{24}

Lookup table

Building a lookup table

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

SMA min: 2

5 – min = 3

00	00	00	03

Delta bytes

		# of values mapping to entry
0	[0 , 0)	1
1	[0 , 0)	1
2	[0 , 0)	1
3	[0 , 1)	1
...		
257	[0 , 0)	2^8
...		
1024	[0 , 0)	2^{24}

Lookup table

i=0

5	20	2	4	290	8	5	420	3	10

Attribute data

Building a lookup table

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

SMA min: 2

290 – min = 288

00	00	01	20

Delta bytes

0x01 + 256*1

\# of values mapping to entry

0	[2 , 3)	1
1	[0 , 0)	1
2	[3 , 4)	1
3	[0 , 1)	1
...		
257	[4 , 5)	2^8
...		
1024	[0 , 0)	2^{24}

Lookup table

i=4

5	20	2	4	290	8	5	420	3	10

Attribute data

Building a lookup table

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

SMA min: 2

5 – min = 3

00	00	00	03

Delta bytes

		# of values mapping to entry
0	[2 , 3)	1
1	[0 , 0)	1
2	[3 , 4)	1
3	[0 , 7)	1
...		
257	[4 , 5)	2^8
...		
1023	[0 , 0)	2^{24}

Lookup table

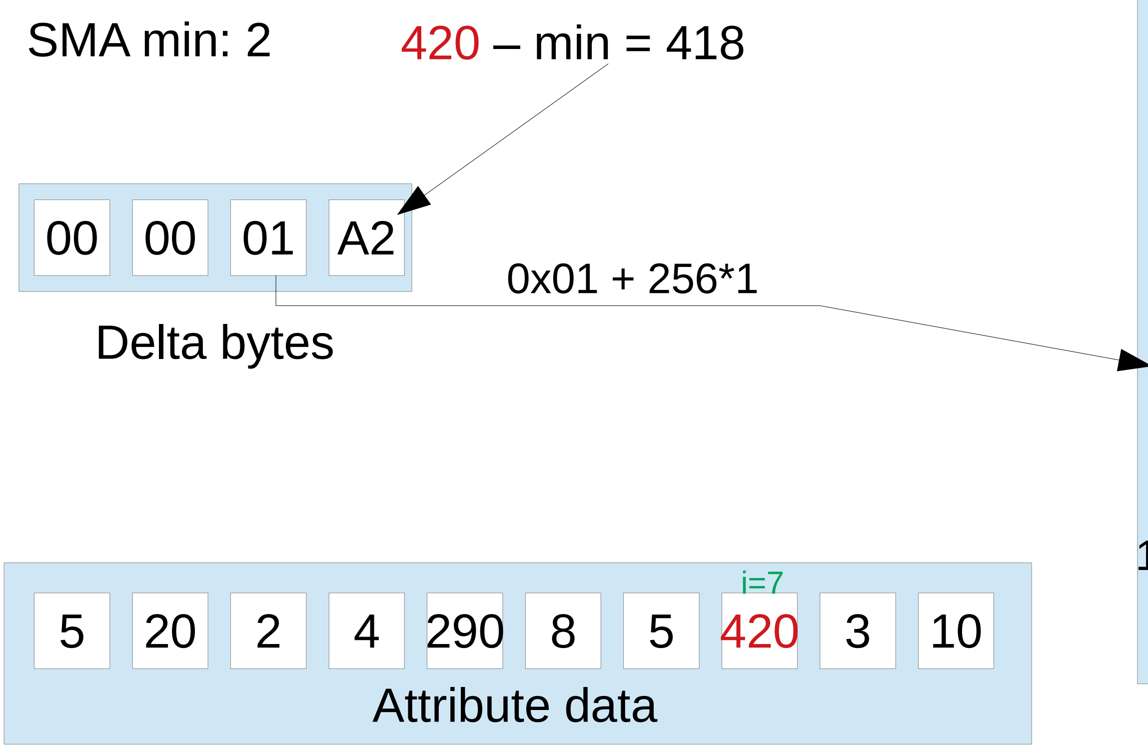
i=6

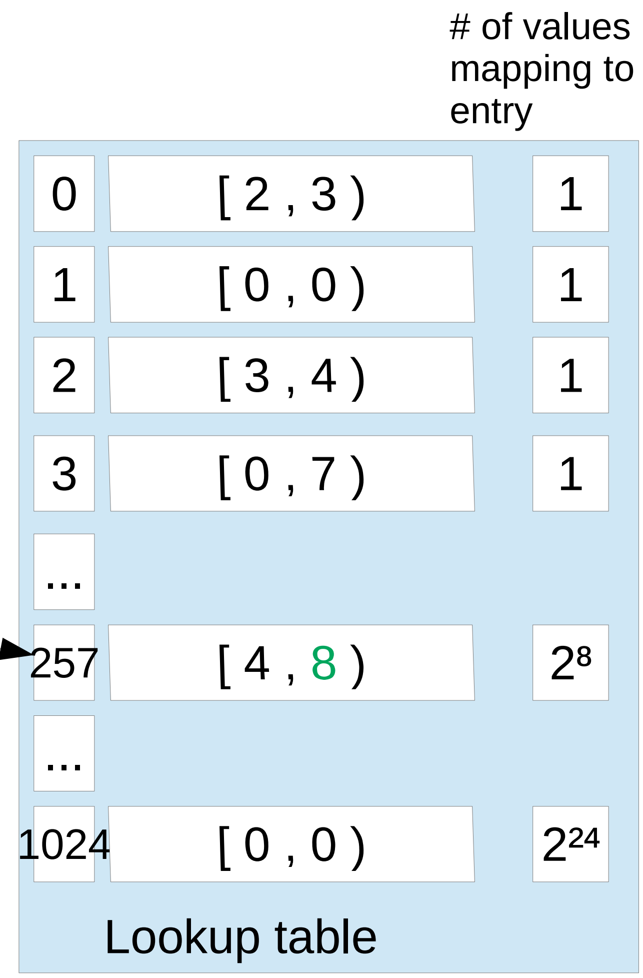
5	20	2	4	290	8	5	420	3	10

Attribute data

Building a lookup table

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





Building a lookup table

- finished lookup table when all data was probed

SMA min: 2

00	00	00	00

Delta bytes

5	20	2	4	290	8	5	420	3	10

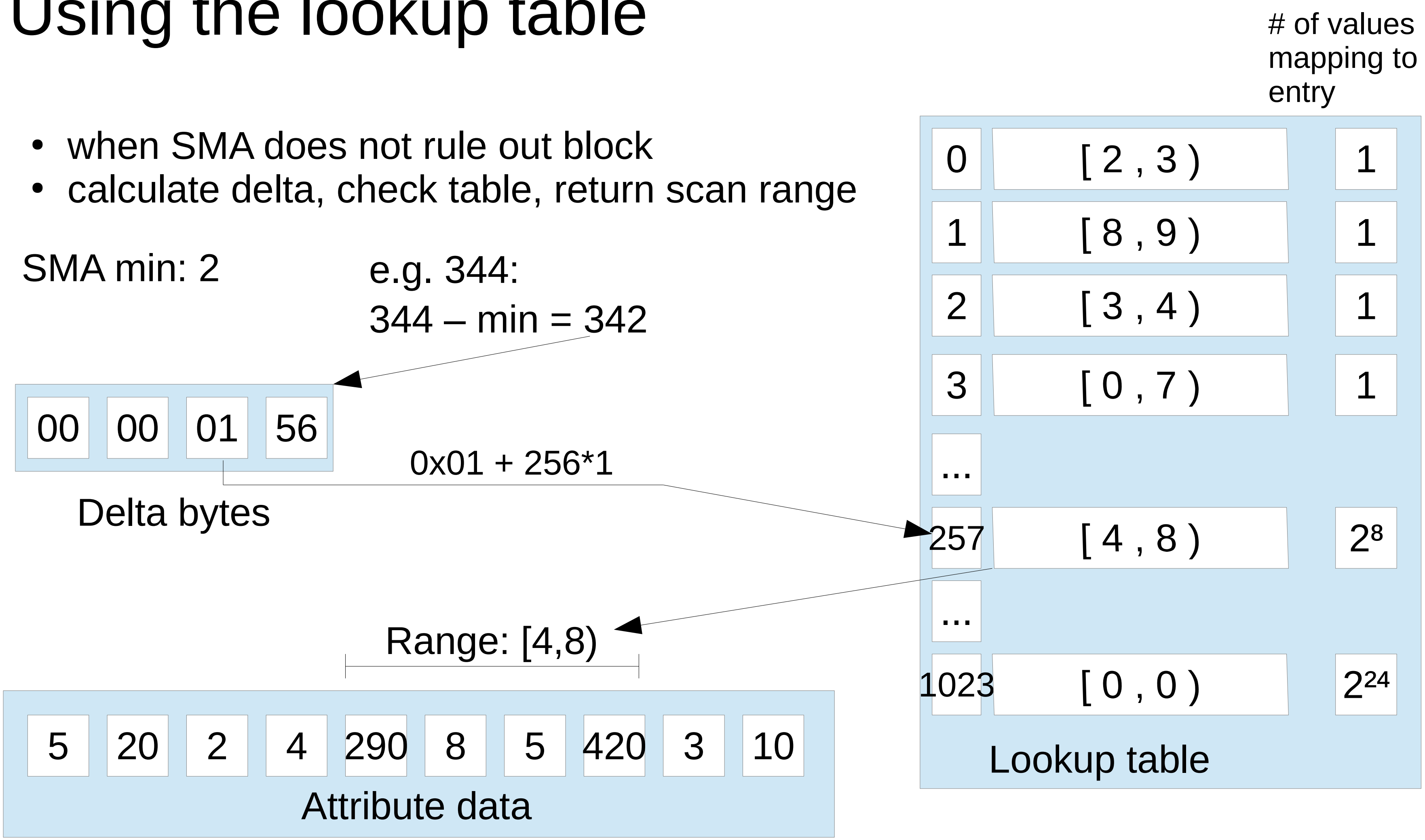
Attribute data

		# of values mapping to entry
0	[2 , 3)	1
1	[8 , 9)	1
2	[3 , 4)	1
3	[0 , 7)	1
...		
257	[4 , 8)	2^8
...		
1023	[0 , 0)	2^{24}

Lookup table

Using the lookup table

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



PSMA

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

Lookup on a Data Block

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

Small overhead for converting restriction constants into compressed version

Further improvements

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

Benchmarks

scan type	geometric mean	sum
JIT uncompressed	0.586s	21.7s
Data Blocks compressed	0.555s (1.06x)	21.5s
+PSMA	0.463s (1.27x)	20.2s
Vectorwise uncompressed	2.336s	74.4s
compressed	2.527s (0.92x)	78.5s

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

Benchmarks

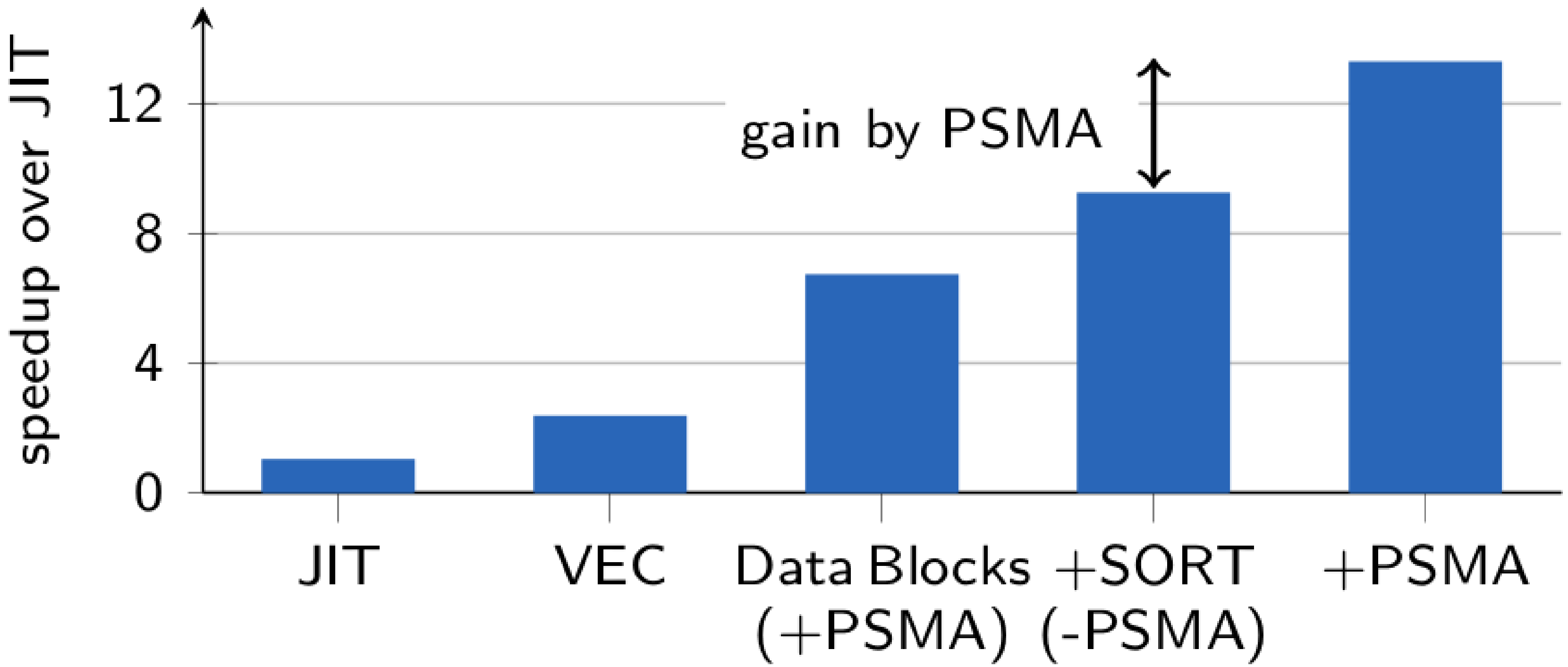
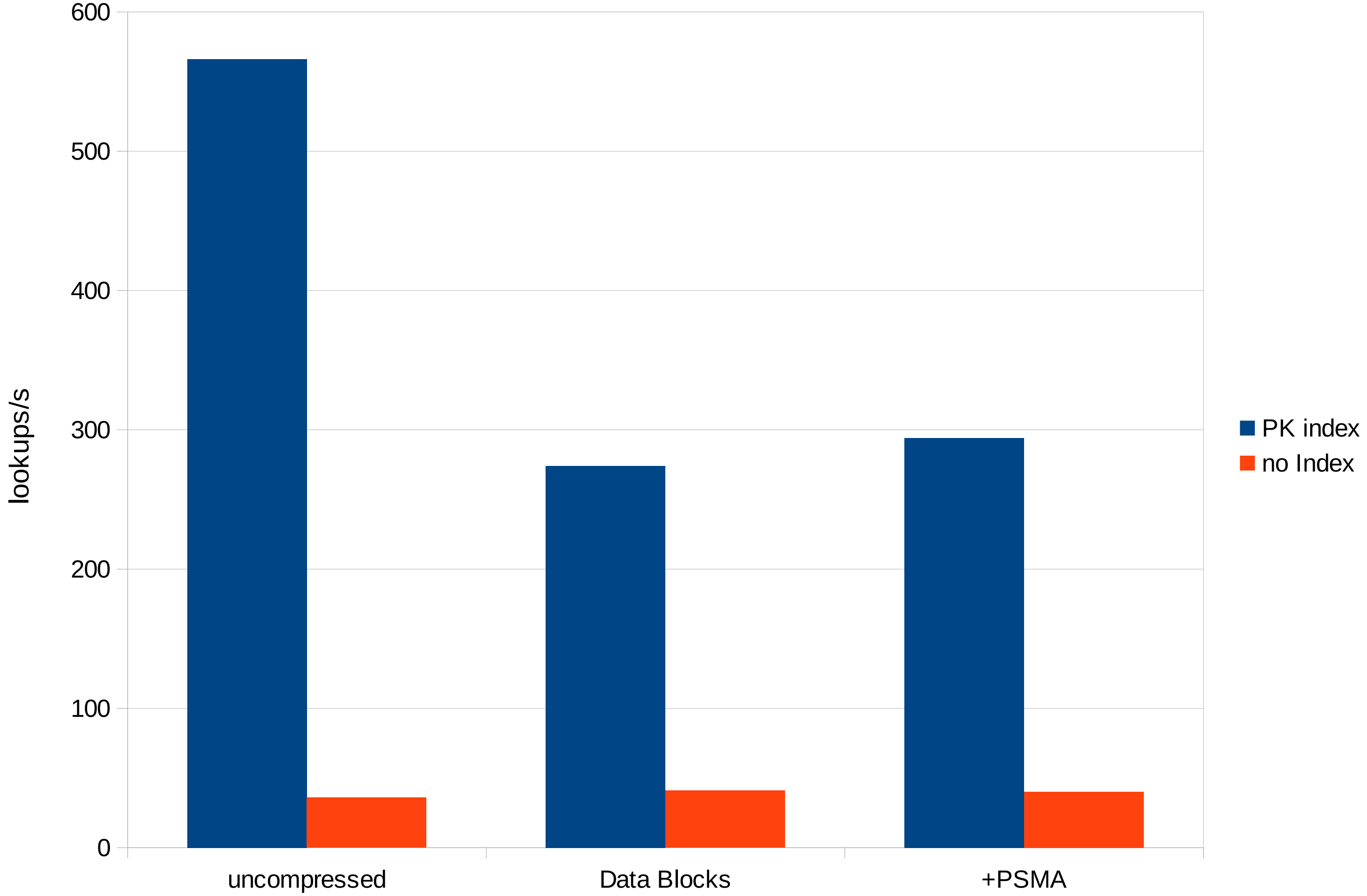


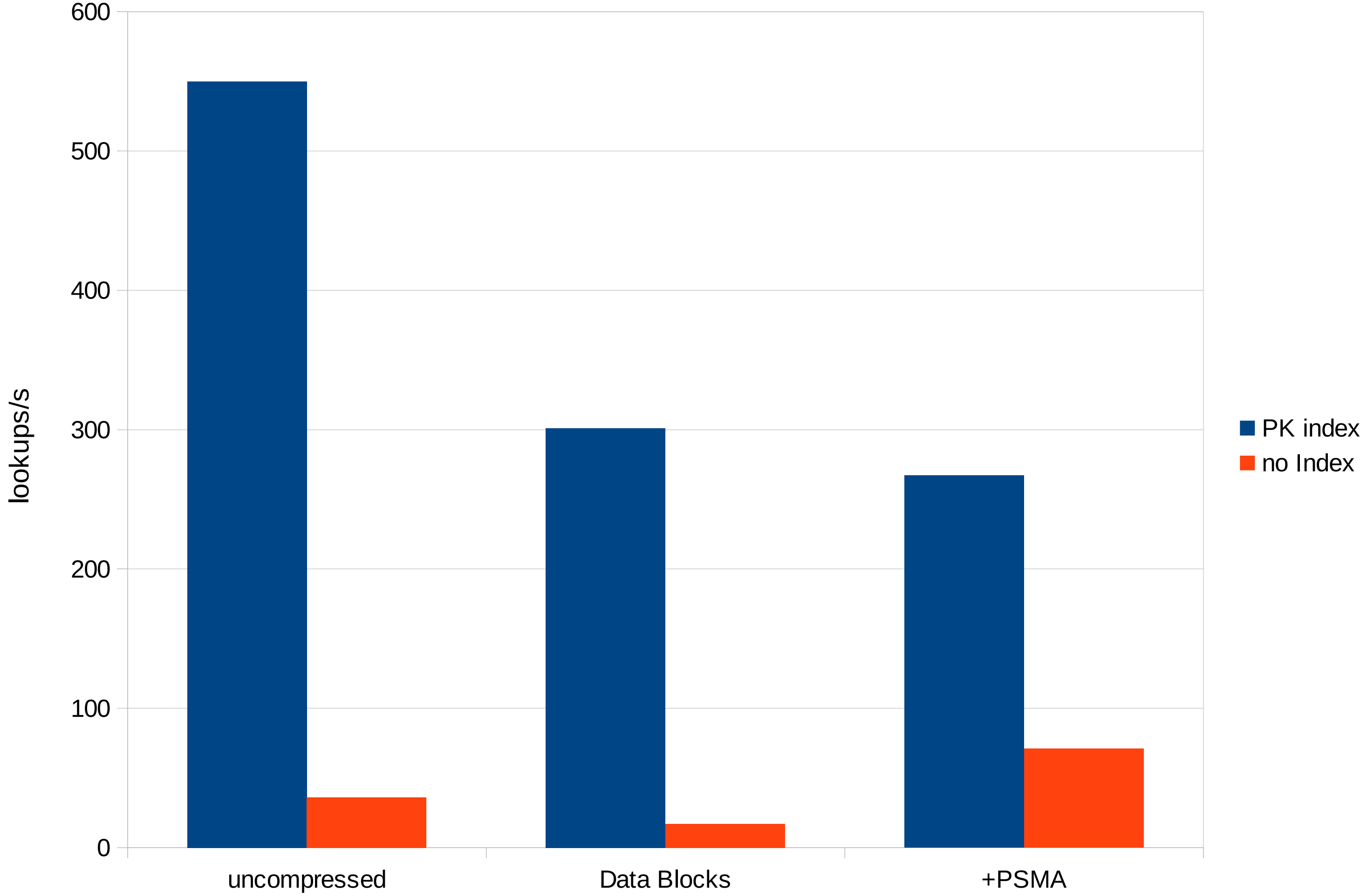
Figure 11: Speedup of TPC-H Q6 (scale factor 100) on block-wise sorted data (+SORT)

[1] Paper: Figure 11

Benchmarks – Shuffled Data



Benchmarks – Ordered Data



Sources

(1) Data Blocks: Hybrid OLTP and OLAP on Compressed Storage using both Vectorization and Compilation

(2) Slides: Data Blocks by Harald Lang