COLUMN STORES

21 OCT 2026

From C-Store to modern analytical databases

Krish Patel
CS848-002 · Modern Database Systems
Professor David Toman



UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

WHY COLUMN STORES EMERGED

Warehouses scan large tables to answer analytical questions

Many queries use only a few columns

CPU work and data movement both limit performance

C-Store explores a database built around this workload

THE WORKLOAD MATTERS

	Transaction example	Analytical example
Question	Update one customer address	Revenue by region for a year
Access	A few records, often many fields	Many records, selected fields
Pressure	Point access and updates	Scanning and aggregation

Stonebraker et al. (2005); Abadi et al. (2008) · workload/layout motivation

ONE TABLE, TWO PHYSICAL LAYOUTS

Row	Region	Amount	Note
1	East	40	A
2	West	70	B
3	East	20	C
4	West	50	D

Row sequence

East, 40, A / West, 70, B
East, 20, C / West, 50, D

Column sequences

Region: East, West, East, West
Amount: 40, 70, 20, 50
Note: A, B, C, D

Illustrative example created for this presentation

A WIDE TABLE MAKES THE COST VISIBLE

3 / 100 columns = 3% of the values

Illustration: 100 equal-width columns, query uses 3

A column scan can leave 97 columns unread

Real I/O also depends on pages, indexes and compression

Illustrative example created for this presentation

A RUNNING ANALYTICAL QUERY

```
SELECT SUM(amount)
FROM Sales
WHERE region = 'East';
```

Matching positions: 1 and 3

Amounts: 40 + 20 = 60

Illustrative example created for this presentation

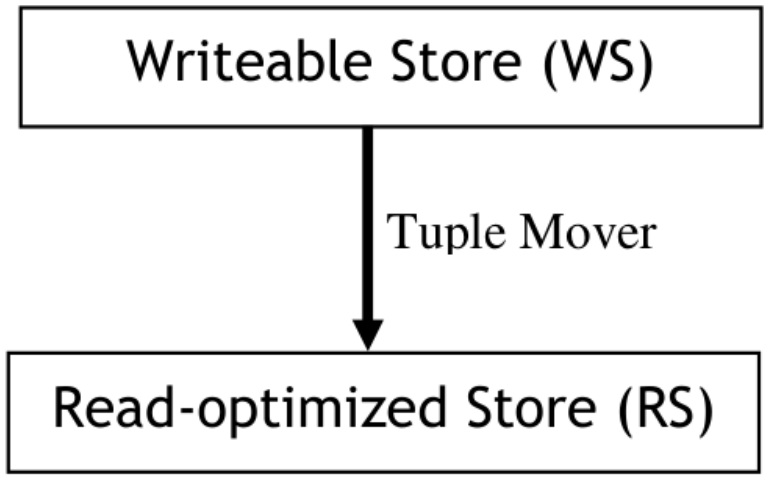
UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

THE THREE PAPERS

Paper	Question
C-Store (2005)	What does a read-oriented architecture look like?
Row vs. column (2008)	Which execution choices explain the gains?
Compression (2006)	How can operators use encoded data directly?

Stonebraker et al. (2005); Abadi et al. (2006, 2008)

C-STORE ARCHITECTURE



SQL over logical tables

Overlapping, sorted projections

Updates enter a writeable store

A tuple mover feeds the read store

Stonebraker et al. (2005) · Figure 1 and §§1–2

SORTED PROJECTIONS

P1 sorted by region

Region	Amount
East	40
East	20
West	70
West	50

P2 sorted by date

Date	Amount
Oct 1	70
Oct 2	20
Oct 3	40
Oct 4	50

Stonebraker et al. (2005) · sorted projections

POSITIONS AND JOIN INDEXES

Logical sale	Position in region order	Position in date order
Sale A	1	3
Sale B	3	1
Sale C	2	2

READS AND WRITES USE DIFFERENT FORMS

Read-optimized store	Writeable store
Compressed columns	Direct, uncompressed values
Order chosen for reading	B-trees maintain logical order
Large, read-oriented component	Smaller component for updates
Receives merged batches	Receives incoming inserts

TRANSACTIONS AND AVAILABILITY

Snapshot reads use epochs and a high-water mark

Replicas can use different physical projections

K-safety targets survival of K node failures

THE OPTIMIZER HAS MORE CHOICES

Choice	Consequence
Projection and sort order	Different access and reconstruction costs
Compressed representation	Different operator and memory costs
Placement of masks	Different intermediate data volumes

Stonebraker et al. (2005) · §8.2; planned cost-based optimizer

WHAT THE FIRST EXPERIMENT SHOWS

Seven simplified TPC-H-style queries

Single-site, read-only measurements

Read-store executor implemented

Writeable store and tuple mover not fully evaluated

Stonebraker et al. (2005) · §9; initial read-only benchmark scope

2005: VIEWS NARROW THE GAP

Query	C-Store	Row baseline	Row + views
Q1	0.03	6.80	0.22
Q3	4.90	93.26	49.38
Q4	2.09	722.90	21.76
Q6	8.50	652.90	47.38

Stonebraker et al. (2005) · §9; selected query times, seconds

ISOLATING THE PERFORMANCE GAP

Experiment	What it tests
Emulate columns inside System X	Does physical partitioning alone suffice?
Remove optimizations inside C-Store	Which execution features matter?

Abadi, Madden & Hachem (2008) · Star Schema Benchmark experiment

EMULATING COLUMNS IN A ROW STORE

Design	Remaining work
Vertical partitions	Repeated tuple headers and position joins
An index on every column	Record-ID lists and index scans
Workload-specific materialized views	Storage and maintenance tied to known queries

Abadi, Madden & Hachem (2008) · row-store emulation alternatives

LATE MATERIALIZATION

Step	Early reconstruction	Late reconstruction
Read	Build candidate tuples	Read predicate columns
Filter	Discard nonmatching tuples	Keep positions 1 and 3
Aggregate	Read surviving amounts	Fetch amounts 40 and 20

BLOCK PROCESSING

An operator consumes a block of values per call

Fewer iterator calls can reduce CPU overhead

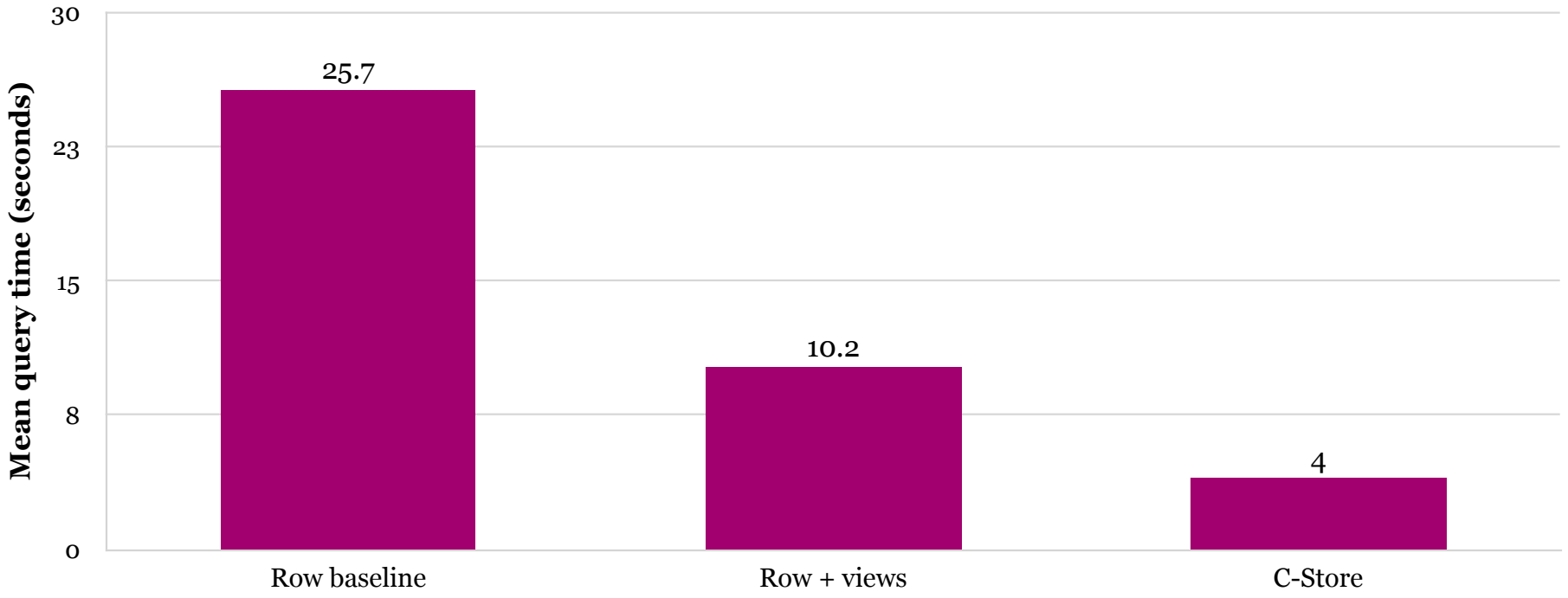
Regular column data supports specialized loops

Abadi, Madden & Hachem (2008) · block processing

INVISIBLE JOINS IN A STAR SCHEMA

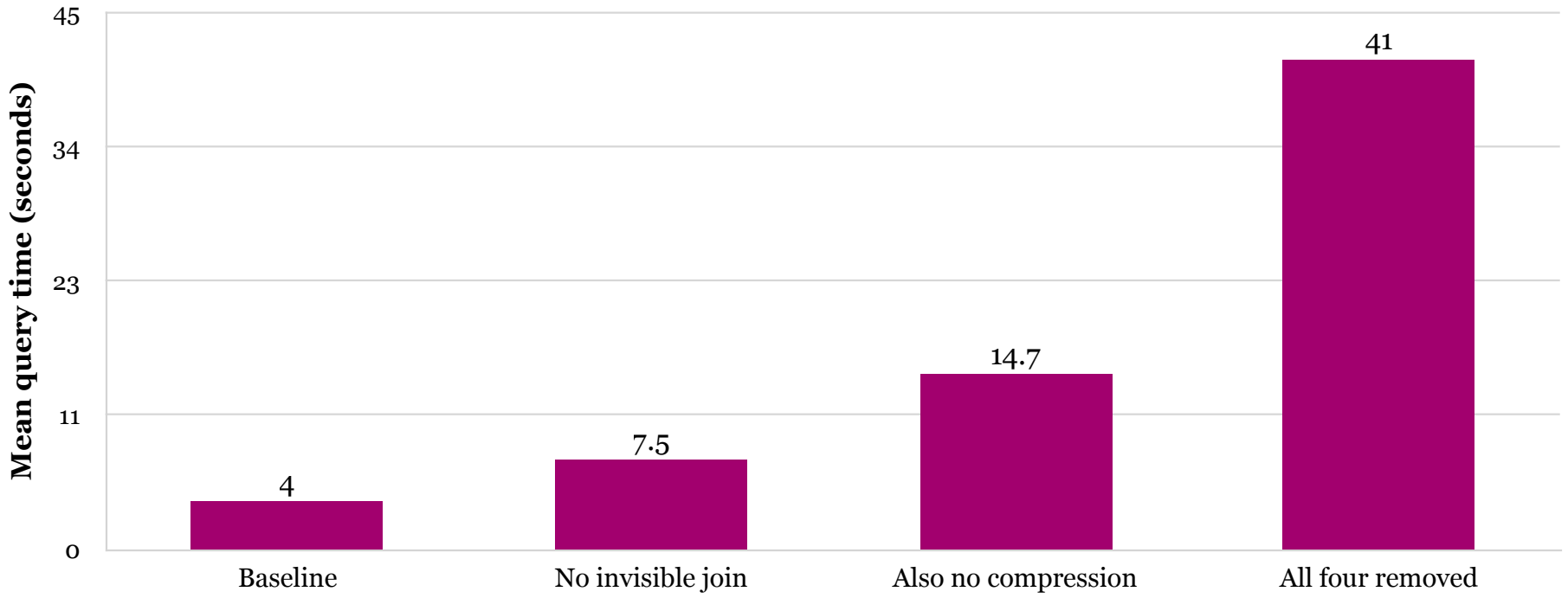
Phase	Operation
1. Dimension filtering	Find qualifying dimension keys
2. Fact filtering	Probe foreign-key columns and intersect positions
3. Value extraction	Fetch grouping and output values for survivors

AVERAGE QUERY TIMES IN THE STUDY



Abadi, Madden & Hachem (2008) · Figure 5; mean across 13 SSBM queries

EXECUTION FEATURES EXPLAIN THE GAP



Abadi, Madden & Hachem (2008) · Figure 7(b); selected configurations

WHAT THE COMPARISON DOES NOT SETTLE

Question	Reason for caution or comparison
All workloads?	Studies emphasize analytical, read-heavy queries
All row stores?	Covering indexes and tailored views can be effective
Layout or execution?	PAX and MonetDB/X100 separate related design choices
Current hardware?	Cache sizes, bandwidth and CPU balance have changed

Abadi, Madden & Hachem (2008) · discussion and related systems

COMPRESSION CHANGES EXECUTION

Fewer bytes can reduce data movement

Decoding can add CPU work

Some encodings let one operation handle many values

ENCODINGS ON A SMALL COLUMN

Representation	Example for East, East, East, West, West
Dictionary codes	East = 0, West = 1 data: 0 0 0 1 1
Run-length encoding	(East, start 1, length 3), (West, start 4, length 2)
Bit vectors	East: 11100 West: 00011

Abadi, Madden & Ferreira (2006) · encoding families; examples adapted for talk

A COMMON API FOR COMPRESSED BLOCKS

Property or operation	Purpose
isOneValue()	A block represents one distinct value
isValueSorted()	Values have a useful sorted order
isPosContig()	Positions form a contiguous range
getNext() / asArray()	Decode values when an operator requires them

Abadi, Madden & Ferreira (2006) · Table 1 and §5

ONE RUN, ONE AGGREGATE

SUM = 7 × 1,000 = 7,000

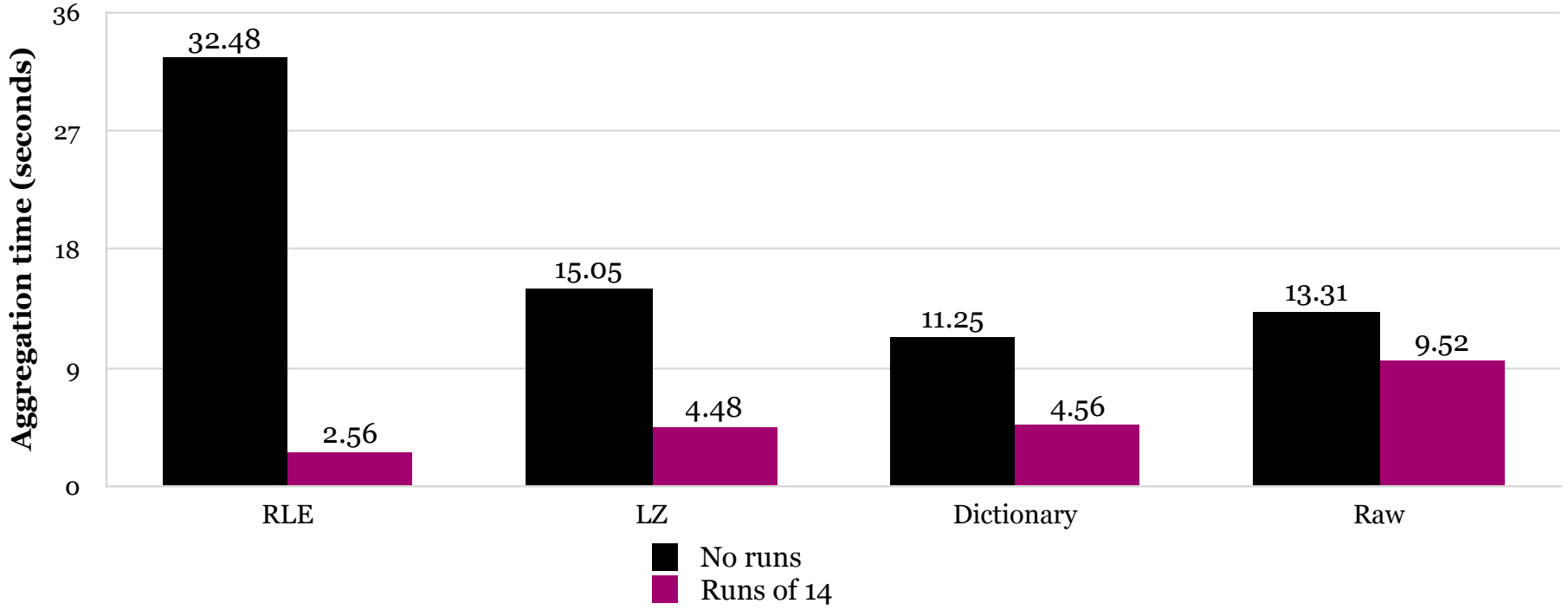
RLE block: value 7, start 1, length 1,000

COUNT over the whole run = 1,000

A partial selection requires the selected run length

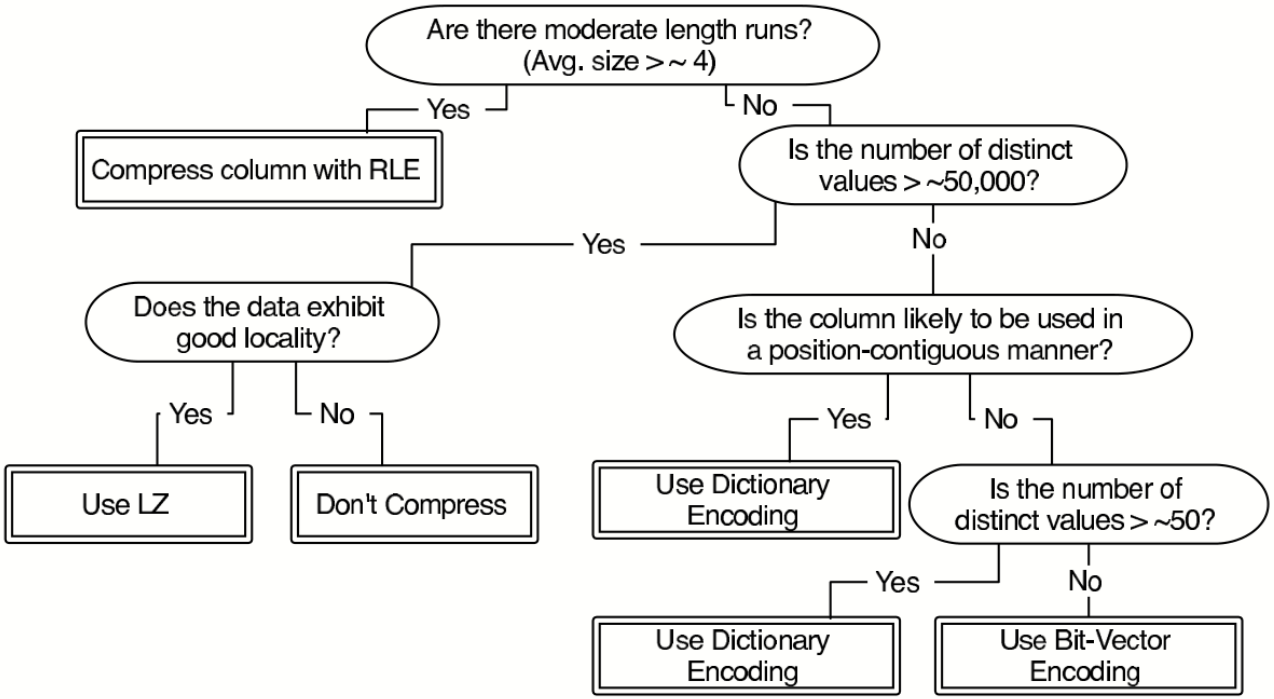
Illustrative RLE example based on Abadi, Madden & Ferreira (2006)

RUNS CHANGE THE BEST ENCODING



Abadi, Madden & Ferreira (2006) · §6.3; mean run length 1 vs. 14

ENCODING CHOICE DEPENDS ON USE



Run length

Domain size

Locality

Position access pattern

Abadi, Madden & Ferreira (2006) · Figure 10; experimental heuristics

C-STORE IDEAS REACH PRODUCTION

System	Connection to this talk
Vertica	Commercialization of C-Store design ideas
Snowflake	Column storage and pruning in a cloud warehouse
DuckDB	Analytical SQL directly over Parquet files
ClickHouse	Deferred column reads for analytical queries

Lamb et al. (2012); Snowflake, DuckDB and ClickHouse documentation

SNOWFLAKE: PRUNING BEFORE SCANNING

Micro-partition	Stored date range	For date = Oct 14
P1	Oct 1–7	Skip
P2	Oct 8–14	Read needed columns
P3	Oct 15–21	Skip

Snowflake documentation · micro-partitions and pruning · checked 20 Sep 2026

DUCKDB: COLUMNS IN OPEN FILES

```
SELECT SUM(amount)
FROM read_parquet('sales.parquet')
WHERE region = 'East';
```

Projection pushdown reads required columns

Filter pushdown can skip row groups using statistics

DuckDB documentation · Parquet projection/filter pushdown · checked 20 Sep 2026

CLICKHOUSE: LAZY MATERIALIZATION

Top-N query stage	Columns needed now
Filter candidate rows	Filter columns
Find the highest-scoring 3 rows	Score and row references
Return the selected reviews	Title, headline and review text

Schreiber (2025), ClickHouse engineering article · lazy materialization

CLICKHOUSE: ONE MEASURED RESULT

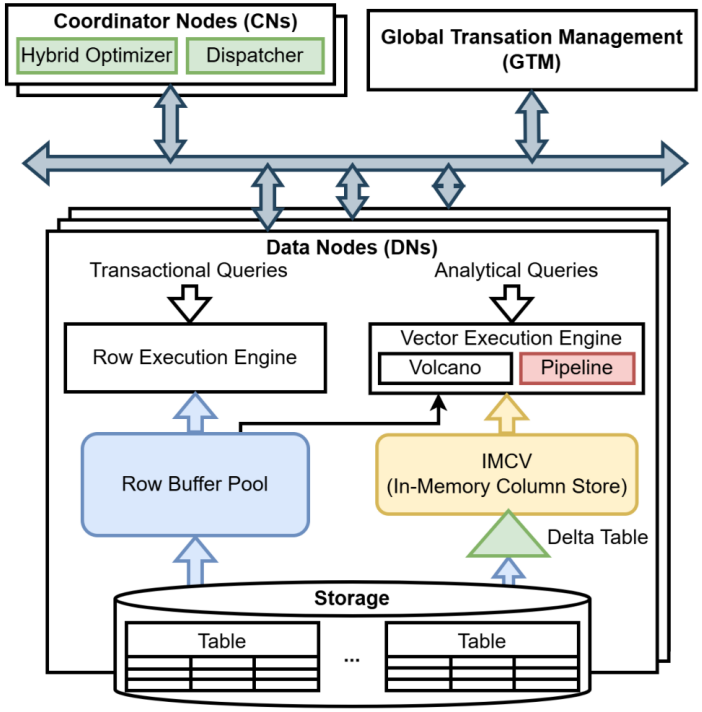
Lazy materialization	Elapsed seconds	Processed data
Disabled	61.148	16.28 GB
Enabled	0.181	807.55 MB

About 151M reviews, cold filesystem cache

32 vCPUs, 128 GiB RAM, gp3 SSD limited to 125 MiB/s

Schreiber (2025), ClickHouse engineering article · developer-reported Top-3 test

2026 RESEARCH: GAUSSDB



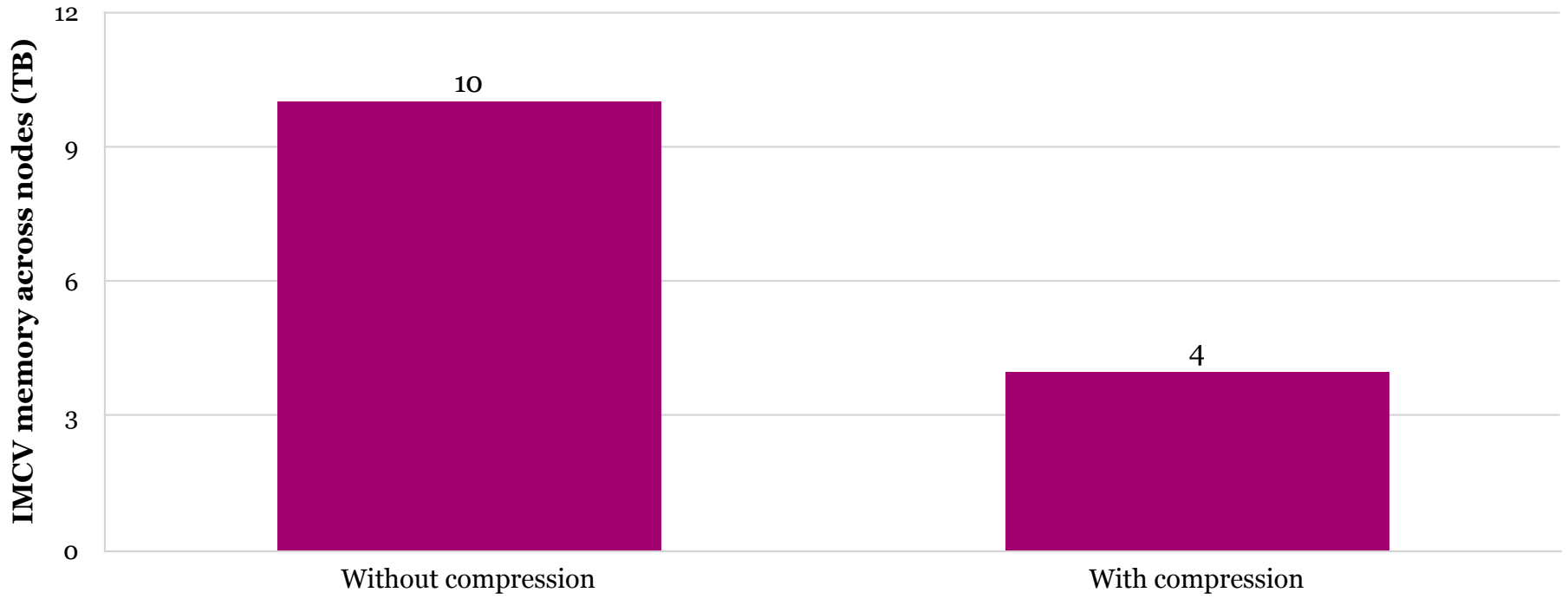
Rows preserve transactional data

An in-memory column view accelerates analytics

Pipelines and distributed filters reduce execution work

Zeyl et al. (2026), PVLDB · Figure 1; HTAP design

GAUSSDB: COMPRESSION SAVES MEMORY



Zeyl et al. (2026), PVLDB · §7.3; 7.5 TB TPC-H memory result

THE LESSON FOR MODERN PLANNING

Read fewer columns and skip irrelevant blocks

Keep useful representations through execution

Include conversion and reconstruction in plan costs

Thesis question: which properties must a planner retain?

REFERENCES: ASSIGNED PAPERS

Stonebraker et al. (2005)
C-Store: A Column-oriented DBMS
VLDB, pp. 553–564

Abadi, Madden and Hachem (2008)
Column-Stores vs. Row-Stores: How Different Are They Really?
SIGMOD, pp. 967–980

Abadi, Madden and Ferreira (2006)
Integrating Compression and Execution in Column-Oriented Database Systems
SIGMOD, pp. 671–682

David Toman et al.
Query Compilation
CS848 course slides

References: assigned papers and CS848 course material

REFERENCES: INDUSTRY AND RESEARCH

Lamb et al. (2012)

The Vertica Analytic Database: C-Store 7 Years Later

PVLDB 5(12), pp. 1790–1801

Snowflake and DuckDB

Micro-partitions & Data Clustering / Reading and Writing Parquet Files

Official documentation, checked 20 September 2026

Tom Schreiber (2025)

ClickHouse gets lazier (and faster): Introducing lazy materialization

ClickHouse engineering article, 22 April 2025

Zeyl et al. (2026)

No Silver Bullet: Boosting GaussDB Performance on the 30TB TPC-H Workload

PVLDB 19(12), pp. 4036–4048

References: industry documentation and 2026 research

QUESTIONS AND DISCUSSION

When can early materialization be cheaper?

Which physical properties belong in planner states?

What would you measure before trusting a speedup?

Sources: see reference slides 38–39