Last Lecture

Eugene Wu

Administrivia

Review session 5/6 2-4PM 480 Mudd

- Thank the staff!

Project 2 due 5/9 10AM. *no late days!*

Extra credit opportunities

- project 2, recursive queries
- write a tutorial
- draw a concept/topic

Grading

Normalize all assignments to [0-1]

Weighed sum based on assignment weights

Compute cut-offs, ~ B+ avg

Then add in extra credit

Agenda

1. Project Winners
2. Modern Databases
3. Q&A / Review

DBMSes in the Wild

Classic Relational

- $$: Oracle, IBM, Microsoft, Teradata, EMC, etc
- Free: MySQL, PostgreSQL

New Relational

- In-Memory, Column-store, Streaming

Non-traditional

- Search (Google, Bing, Lucene), Scientific, Geographic

NoSQL

- Big Data: Hadoop, Spark, etc
- Key-value: Mongo, BerkeleyDB, Cassandra, etc

DBMS-as-a-Service

- Microsoft Azure, Amazon Redshift/RDS, etc…

DBMSes in the Wild

Classic Relational

- $$: Oracle, IBM, Microsoft, Teradata, EMC, etc
- Free: MySQL, PostgreSQL

New Relational

- In-Memory, Column-store, Streaming

Non-traditional

- Search (Google, Bing, Lucene), Scientific, Geographic

NoSQL

- Big Data: Hadoop, Spark, etc
- Key-value: Mongo, BerkeleyDB, Cassandra, etc

DBMS-as-a-Service

- Microsoft Azure, Amazon Redshift/RDS, etc…

Modern Database Systems

Hardware changes affect

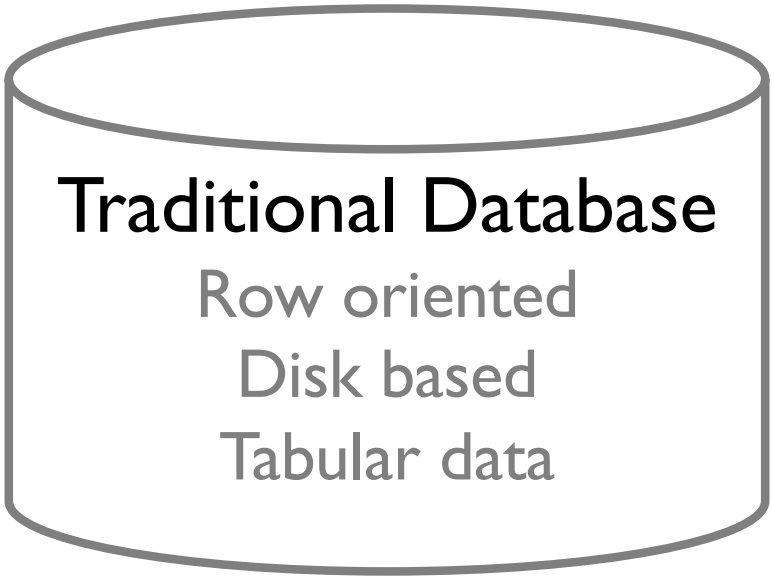
Compression is good → Column stores

- Large scale aggregation queries
- "Data Warehouses"

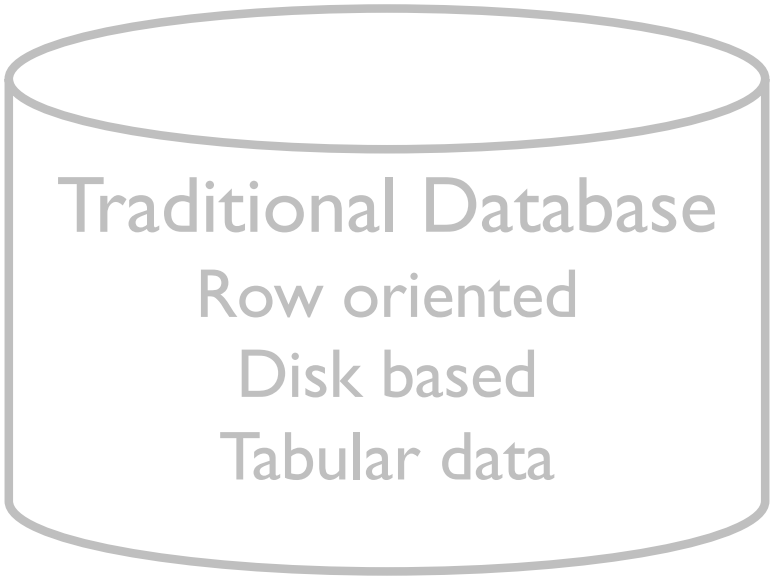
Memory is cheap → In-memory stores

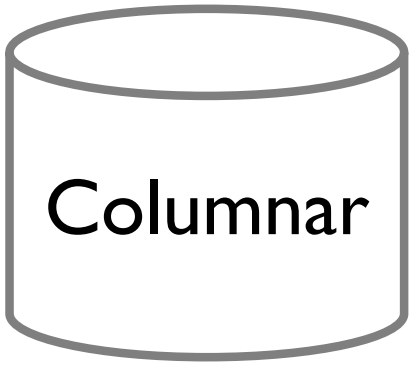
- Transactional systems

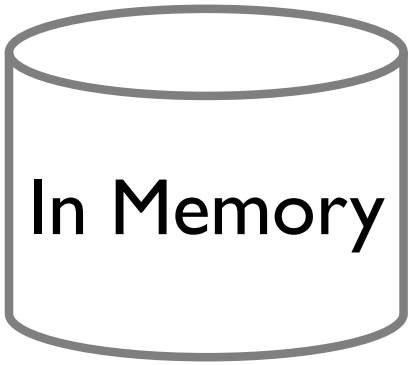
One Size Fits All

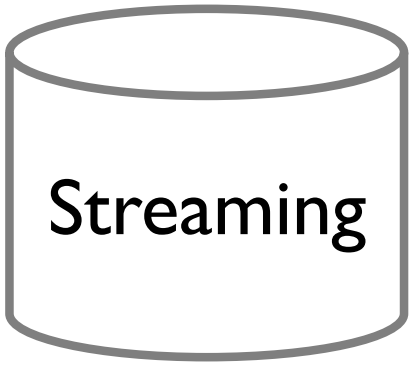


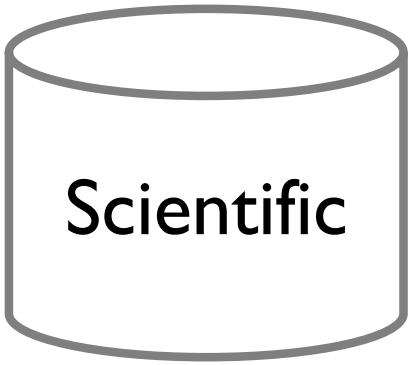
One Size Does Not Fits All



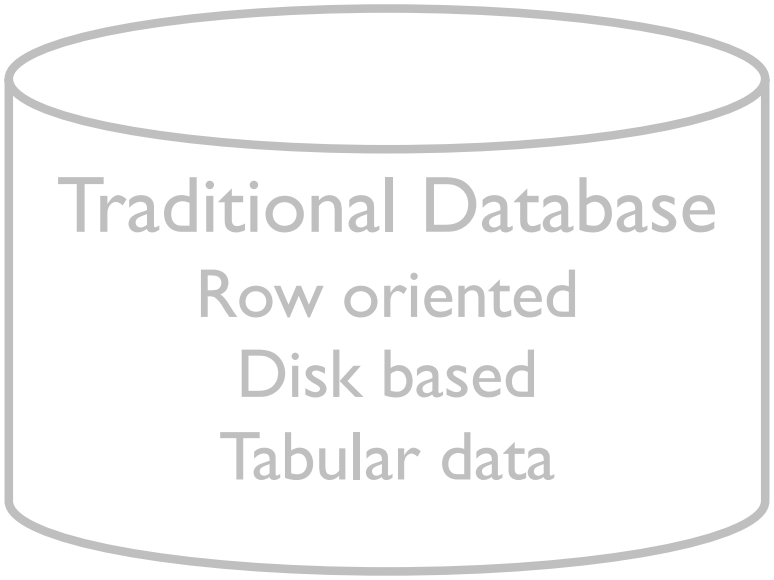


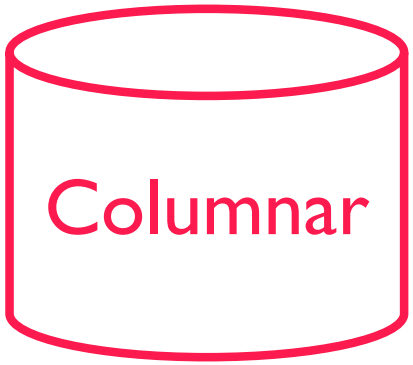


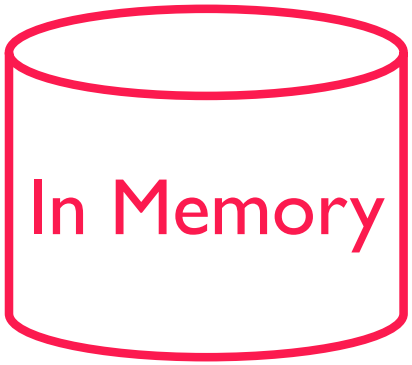


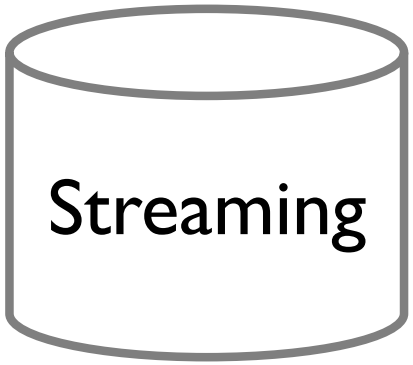


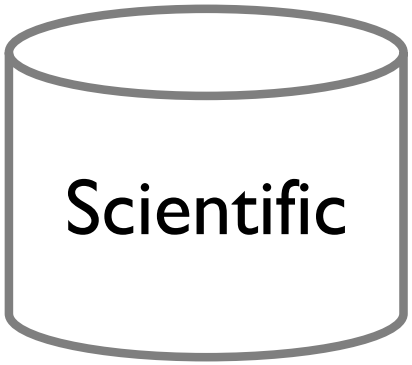
One Size Does Not Fits All











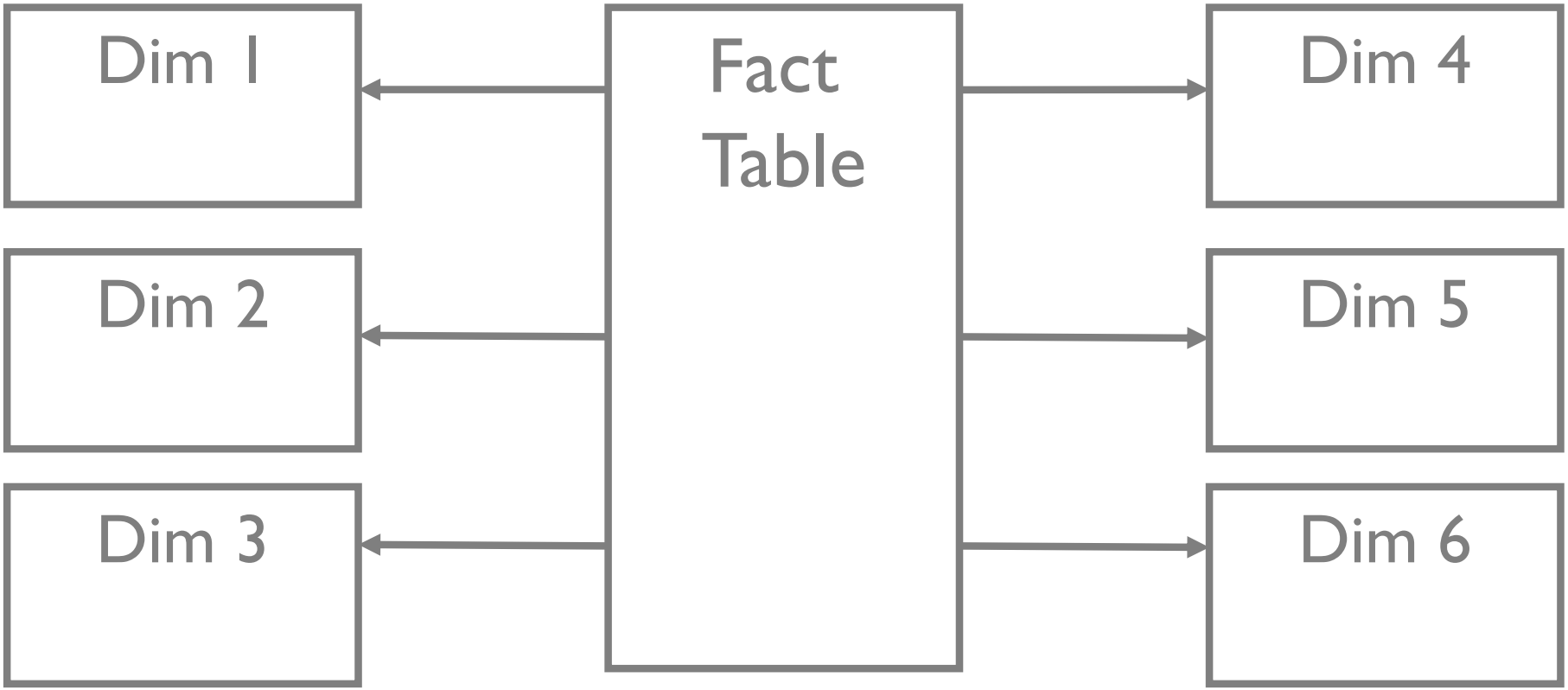
Data Warehouses

Store all historical data for future analysis

- Sales by month over past 20 years
- Clicks by youth in texas
- Cost by product component

Defacto standard at any company

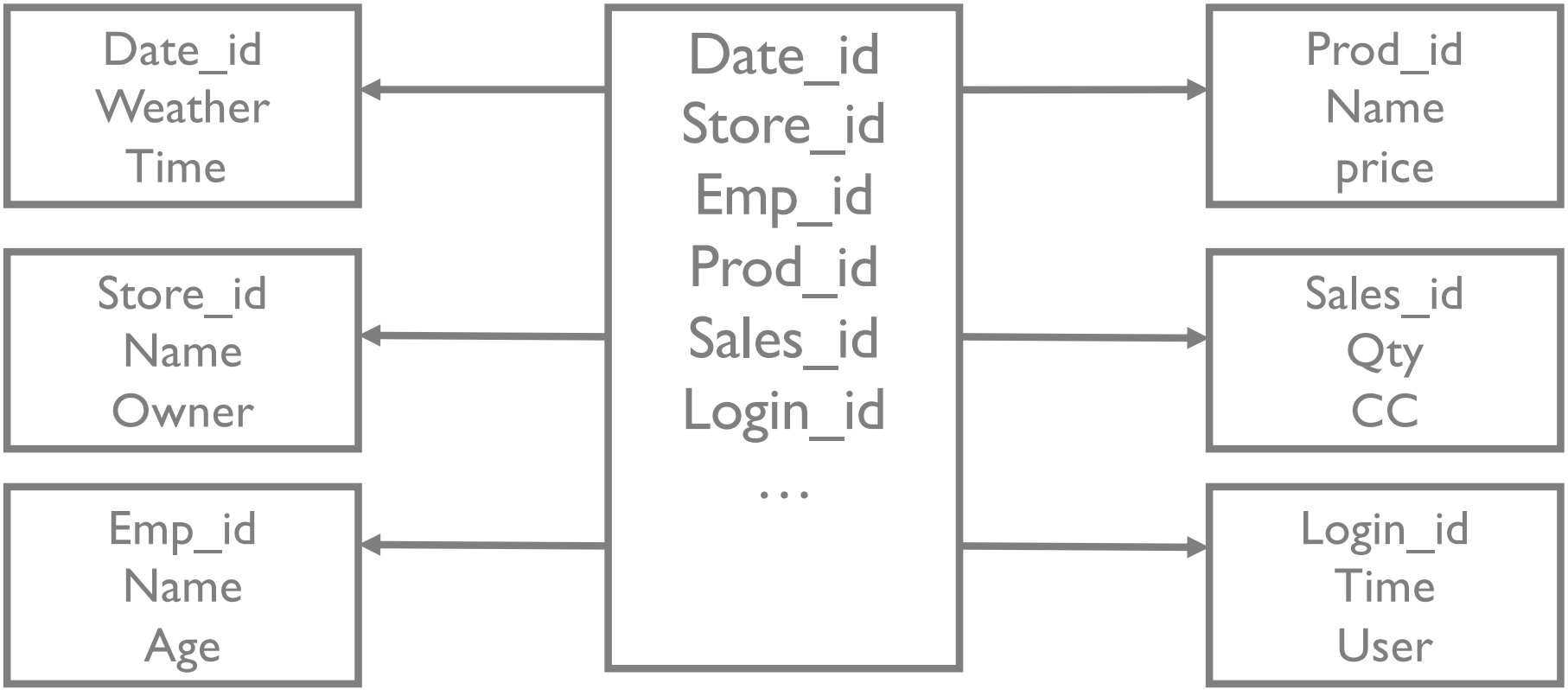
Star Schema

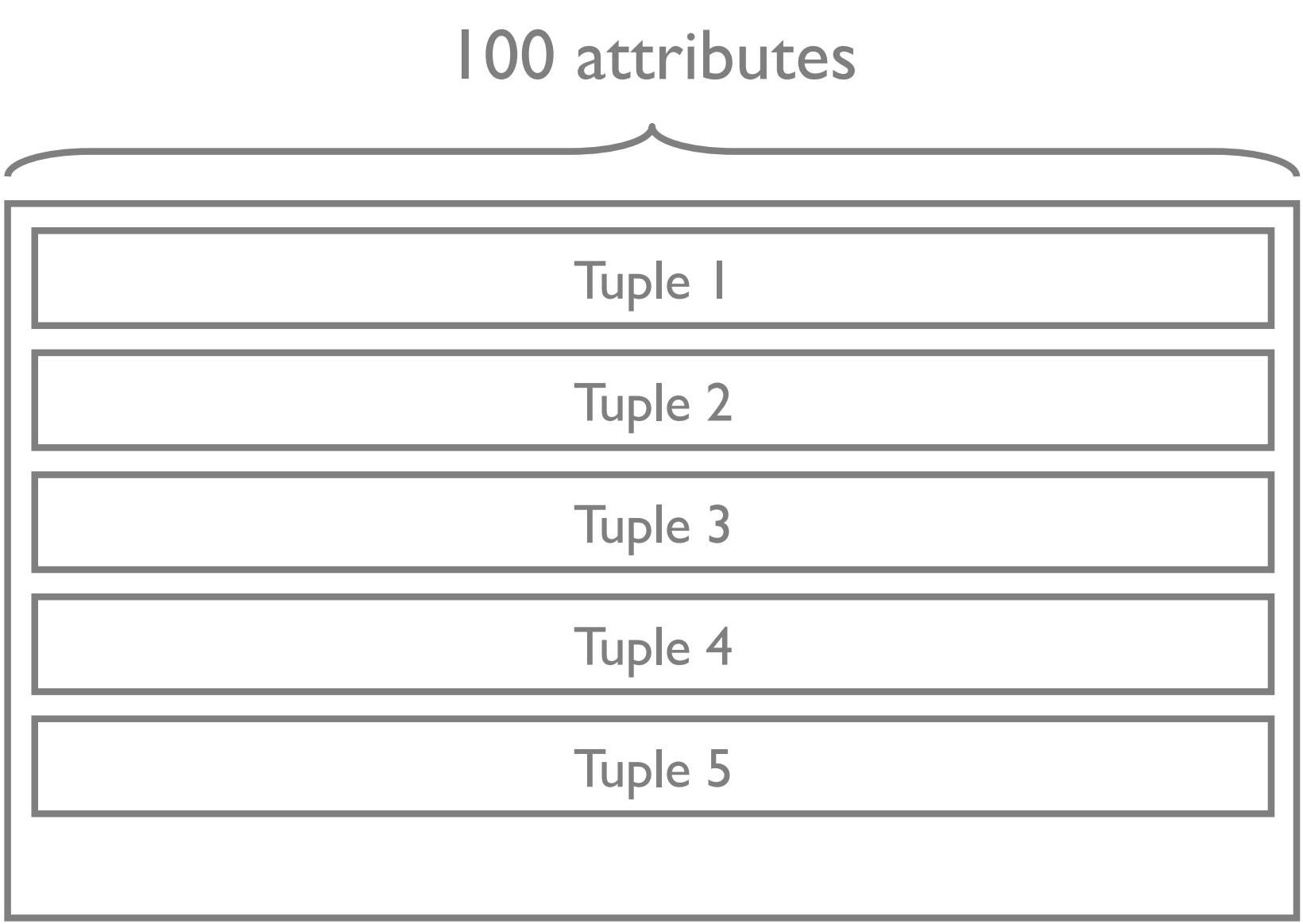


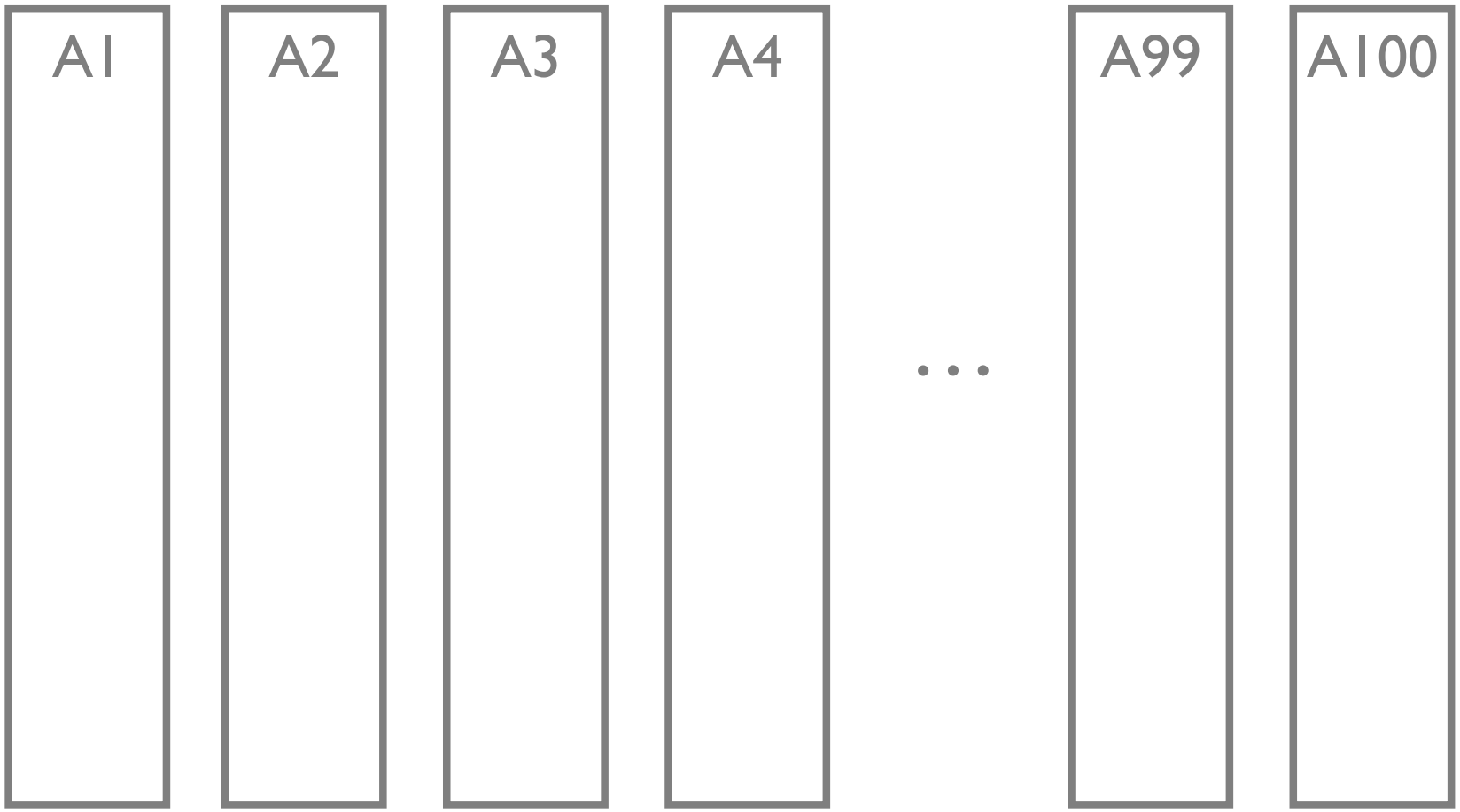
Star Schema

Fact table is “fat”

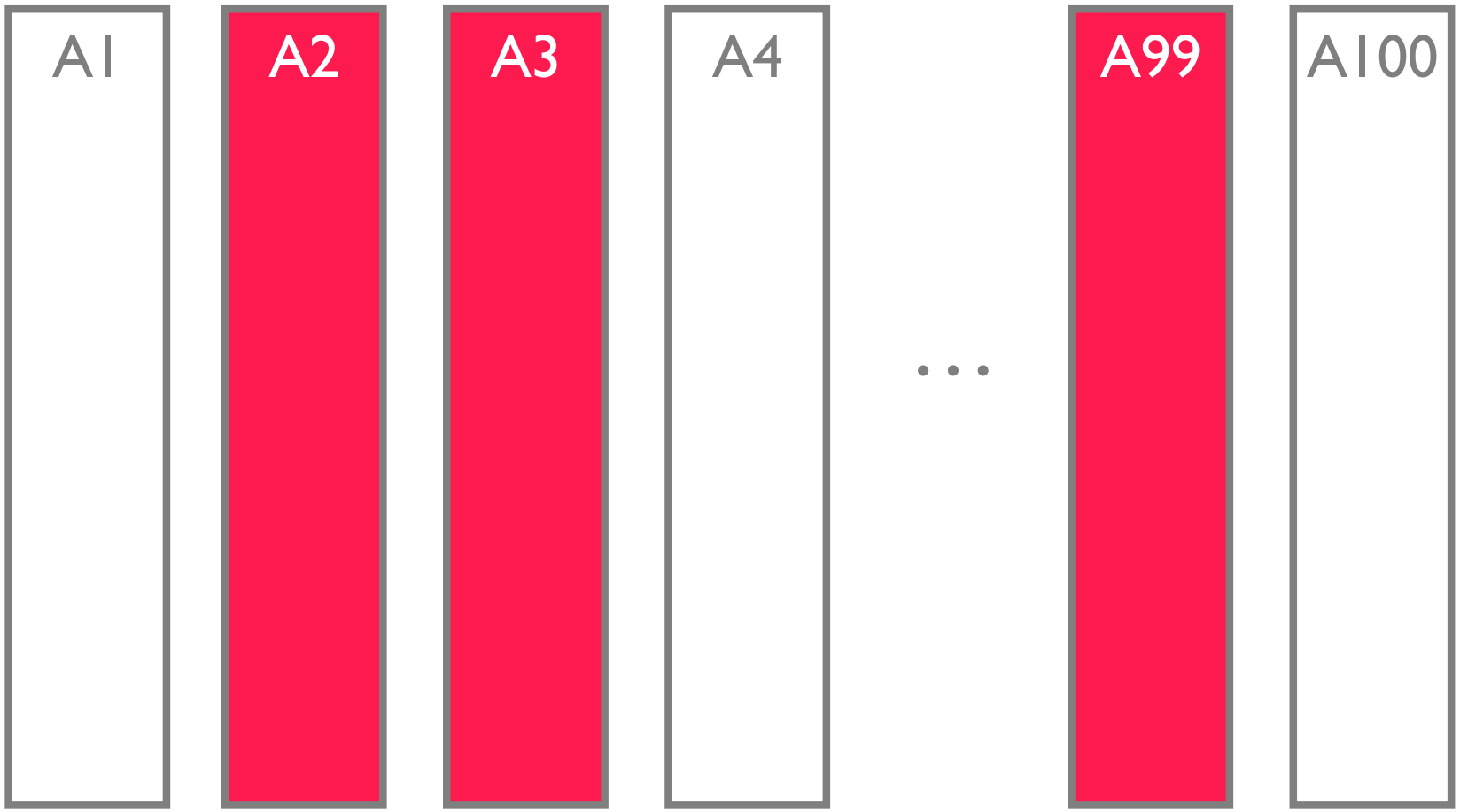
Queries access ~6 attrs




100 attributes
Tuple 1
Tuple 2
Tuple 3
Tuple 4
Tuple 5


A1
A2
A3
A4
A99
A100

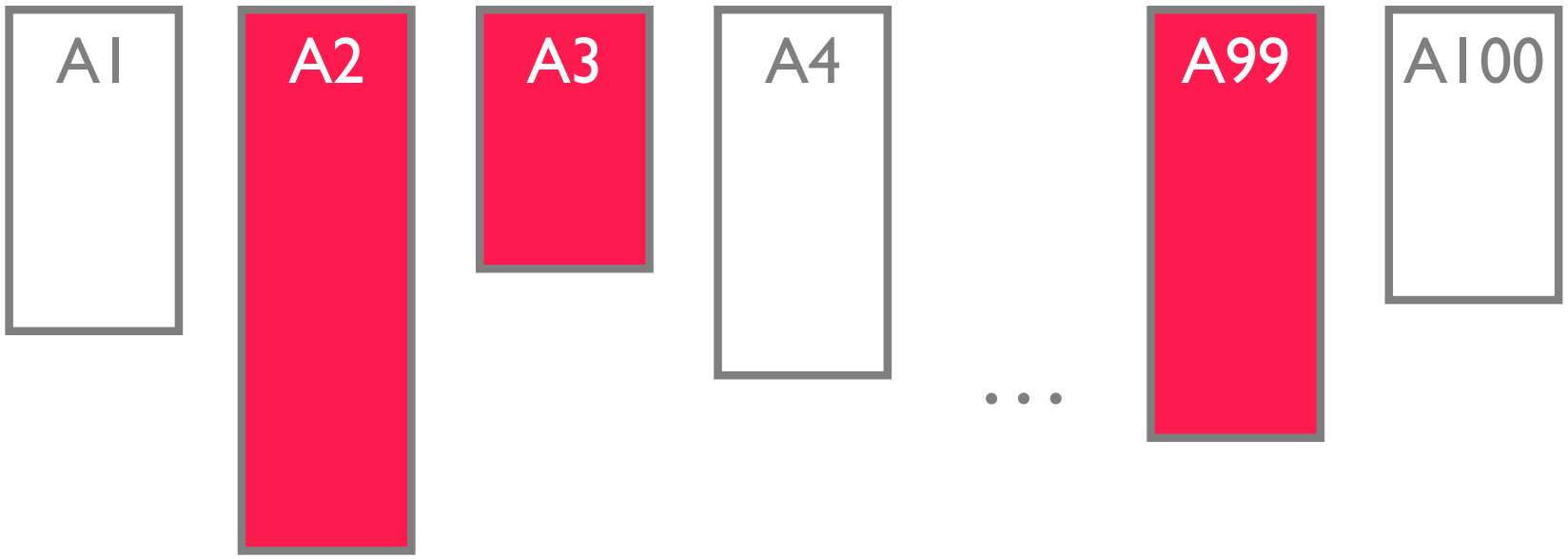
16x less data read. Unfair advantage.



16x less data read. Unfair advantage.

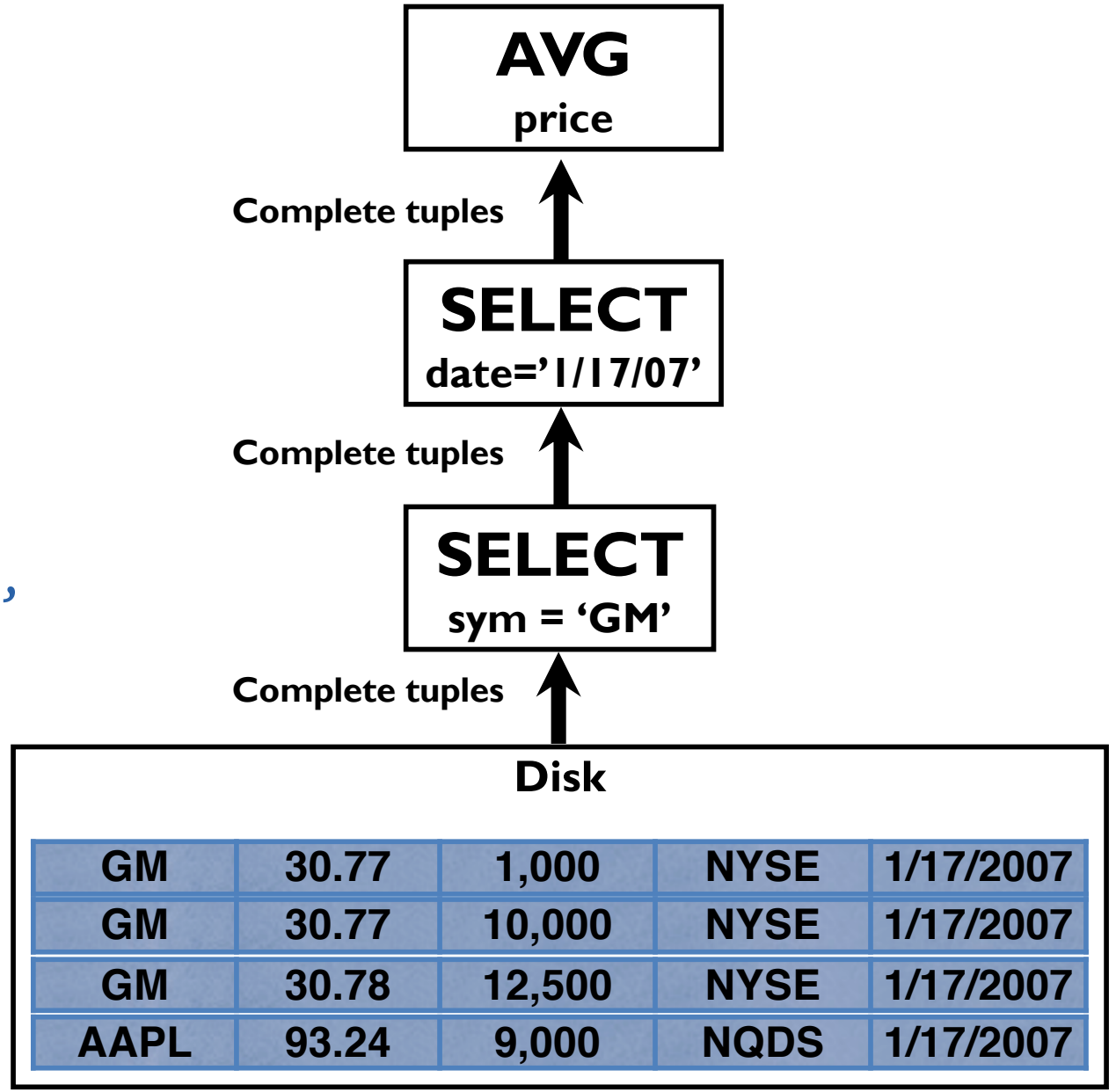
Compression better on single column

Execute on compressed data



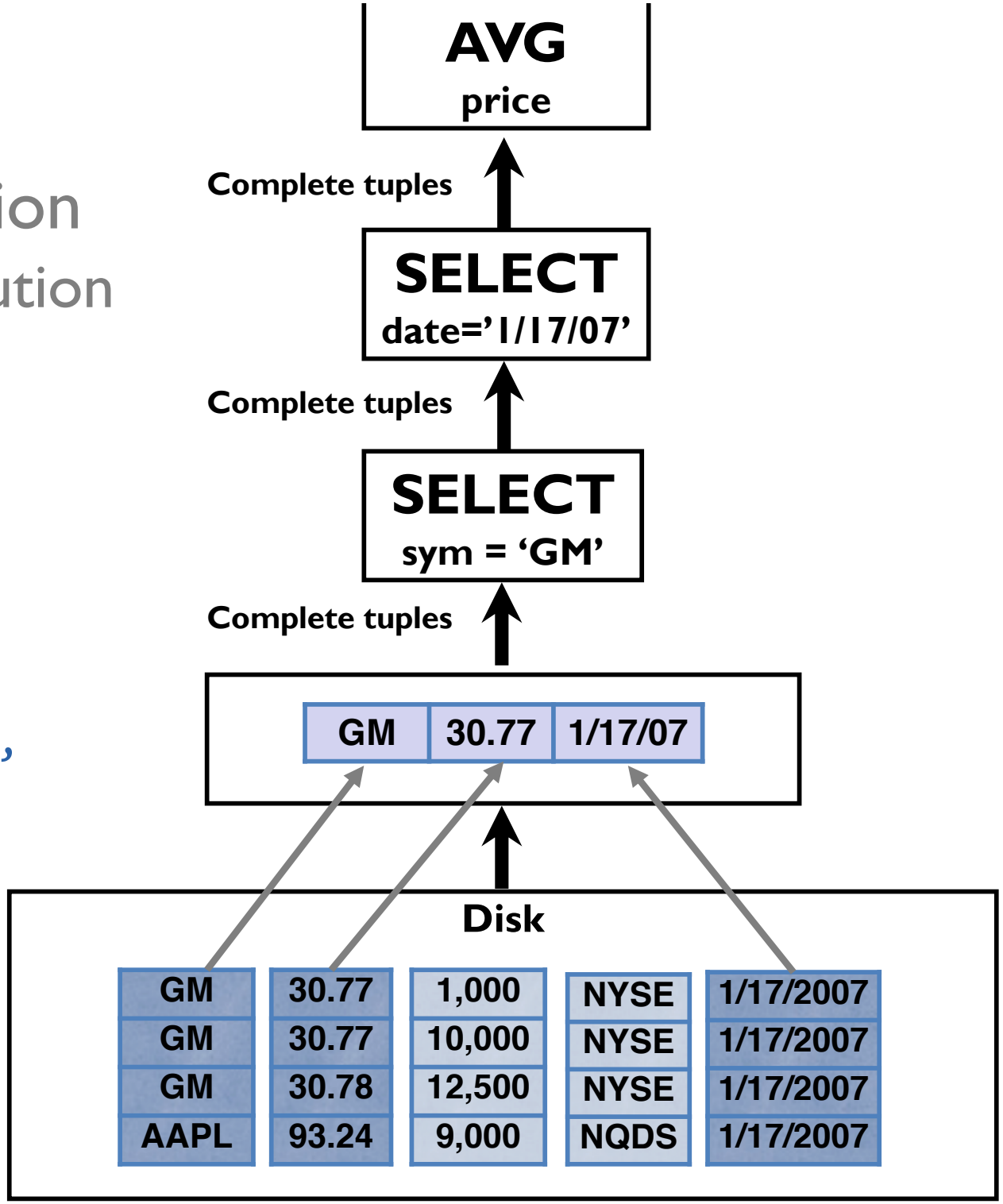
Traditional DBMS

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



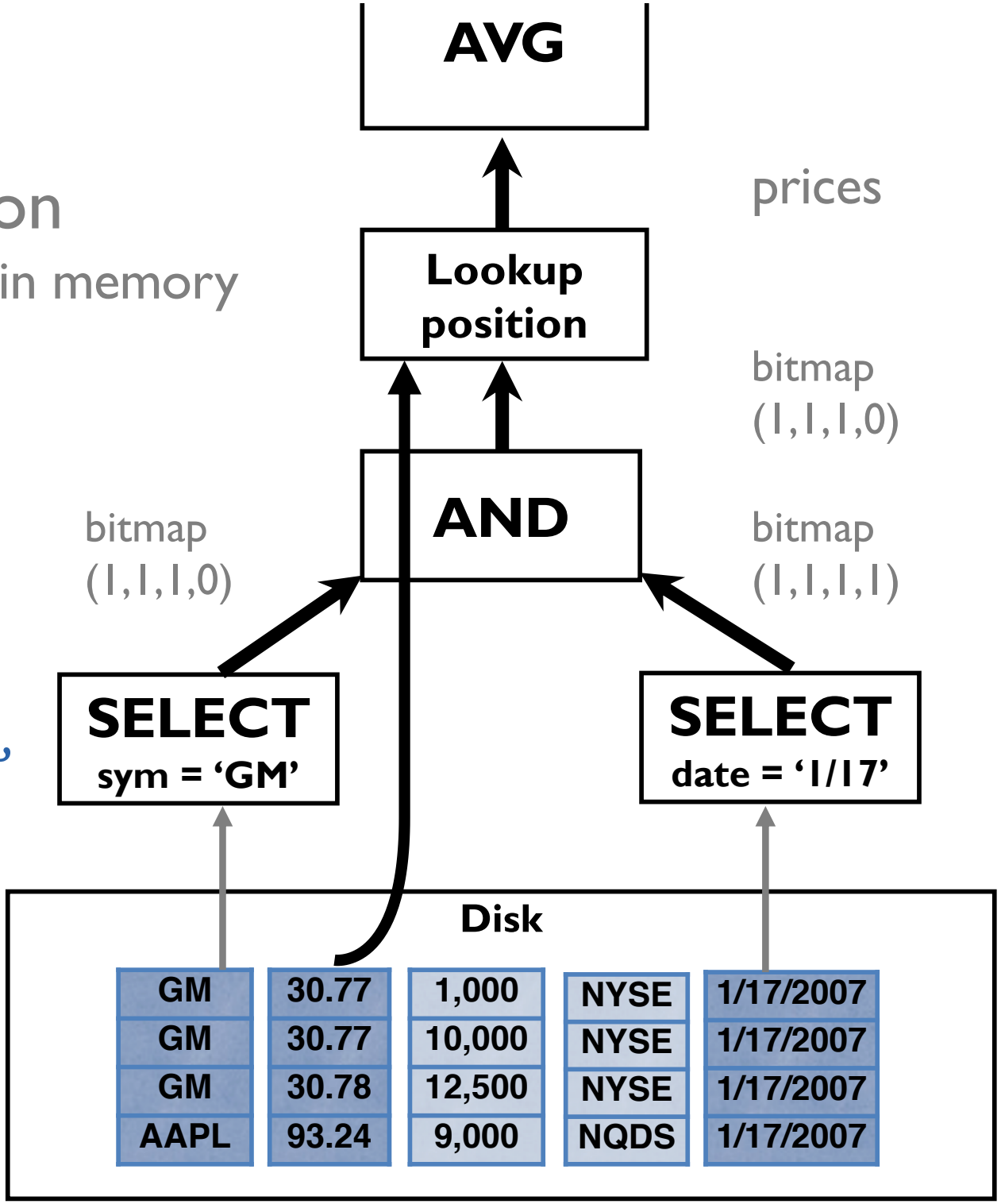
Naïve:
Early Materialization
Row oriented execution

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



C-Store
Late Materialization
Much less data moving in memory

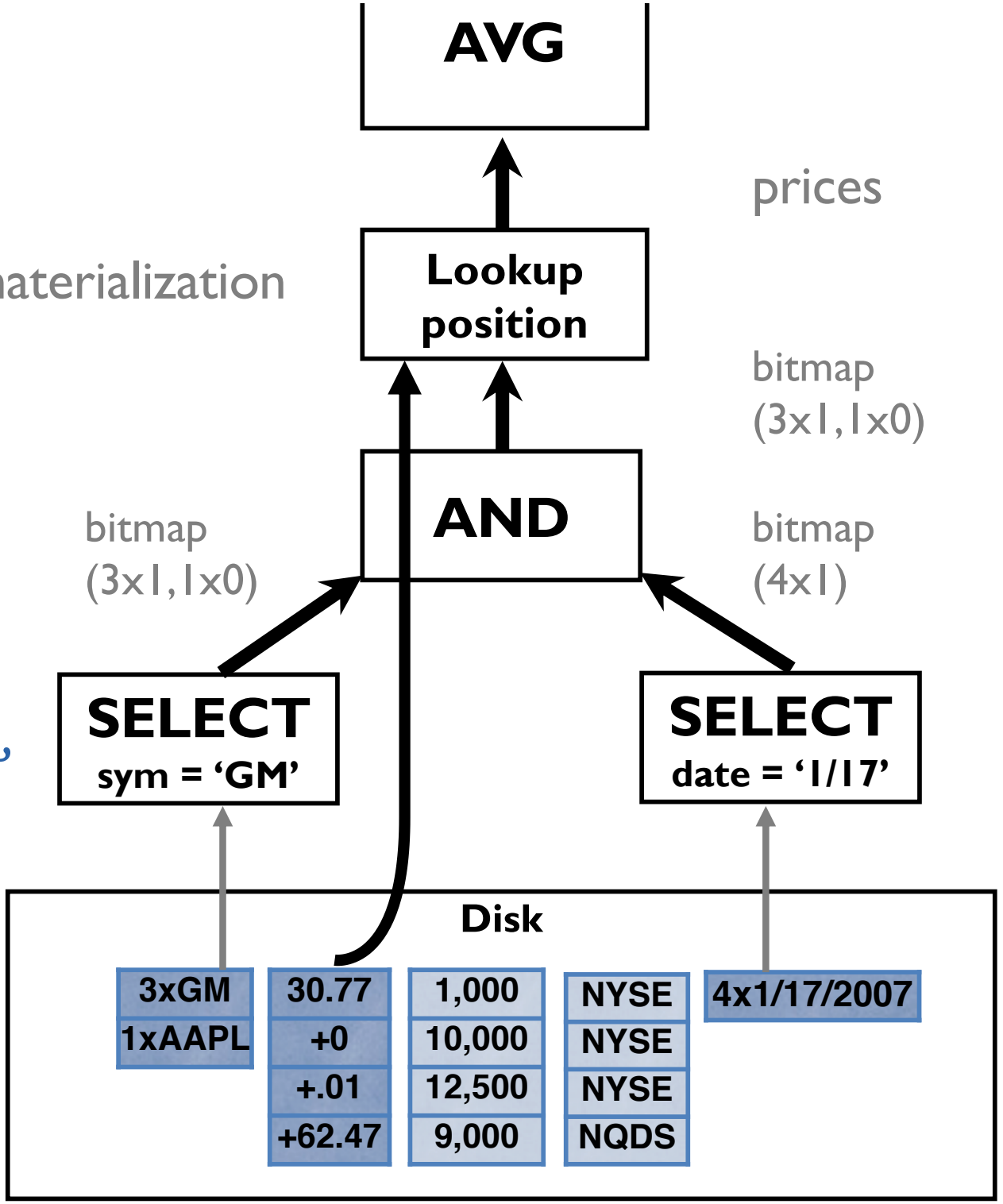
```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



C-Store Compression

Only possible w/ late materialization

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



Column Stores

Optimized for data warehouses

Store data by attribute/column rather than row

Compression

Compressed *query plan execution*

50-100x faster than row store

In-Memory DBMSes

Transaction-oriented apps

- remove 1 unit from product
- move 5 units from org 1 to org 2
- (shopping carts, inventory)

Data stored in memory

- Disk only used for check pointing

No concurrency

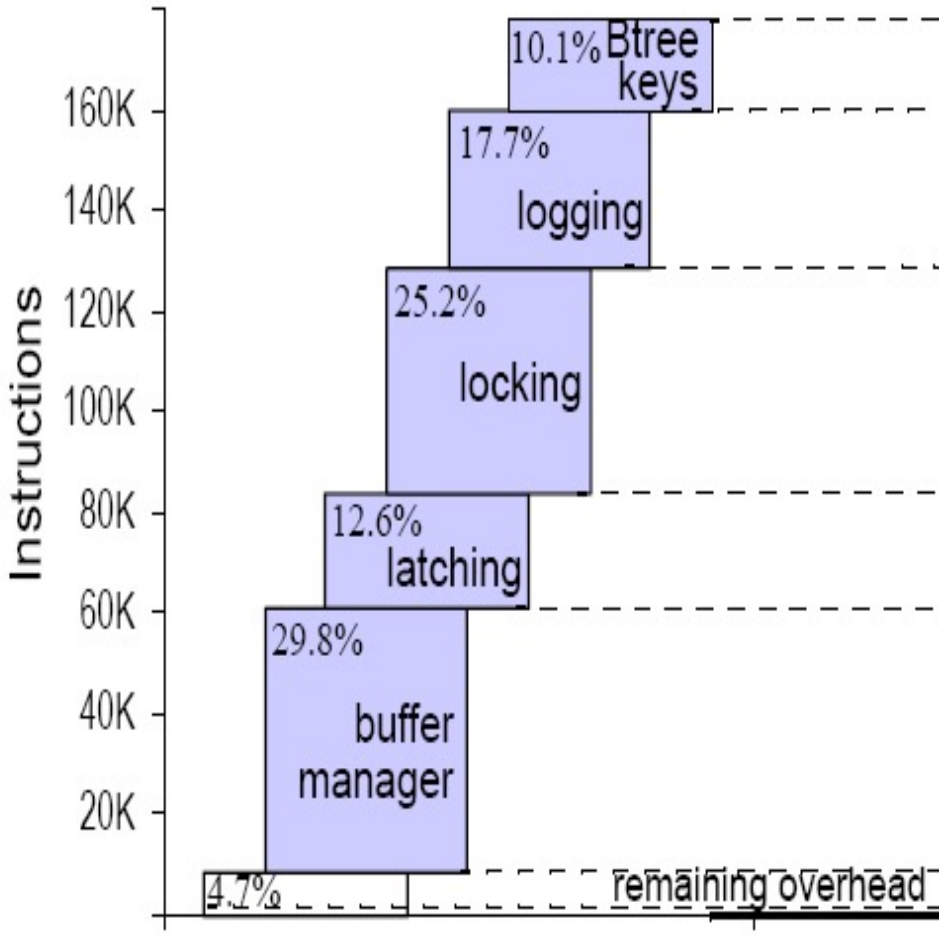
Active-active replication for fault-tolerance

Traditional Database

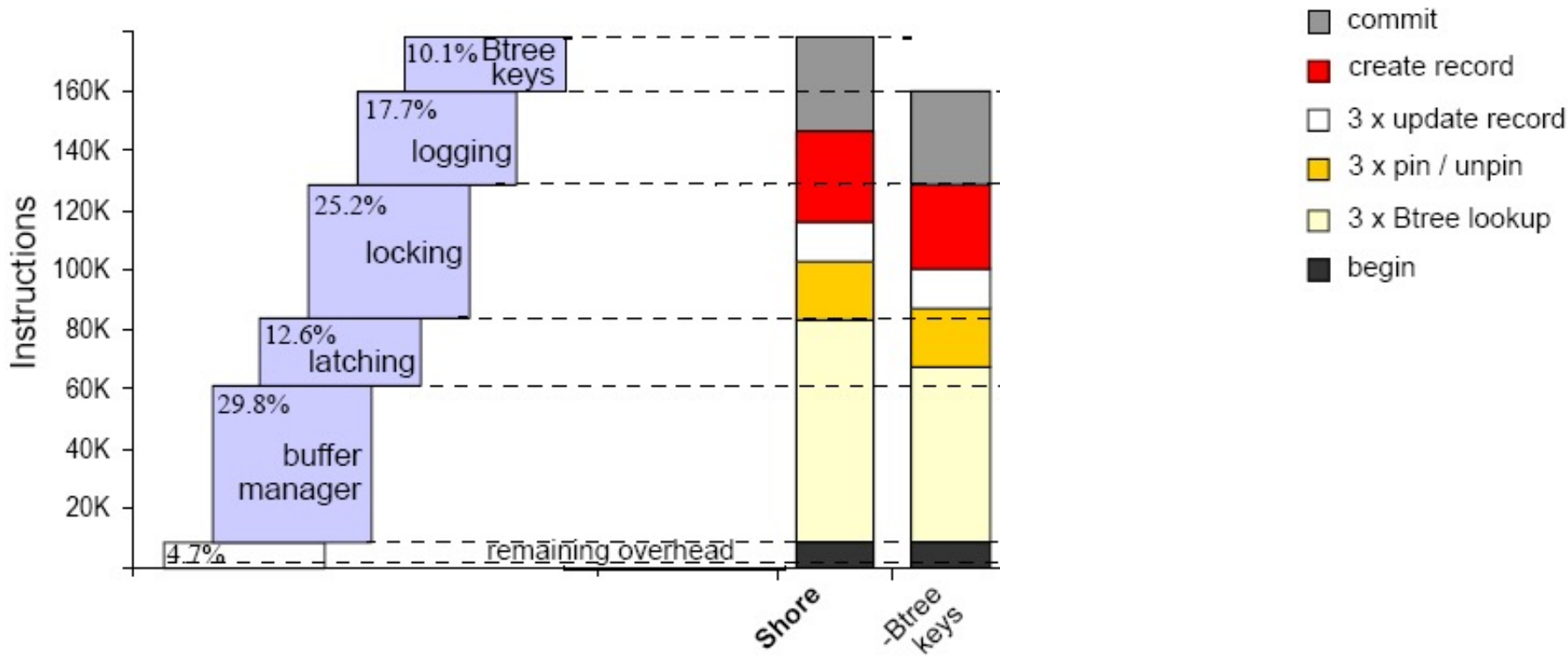
Indexes	queries go faster
Concurrency	queries go faster
Locking	serializability
Logging	recovery
Buffer Manager	manage pages in memory

Results after removing the components (in # instruction)

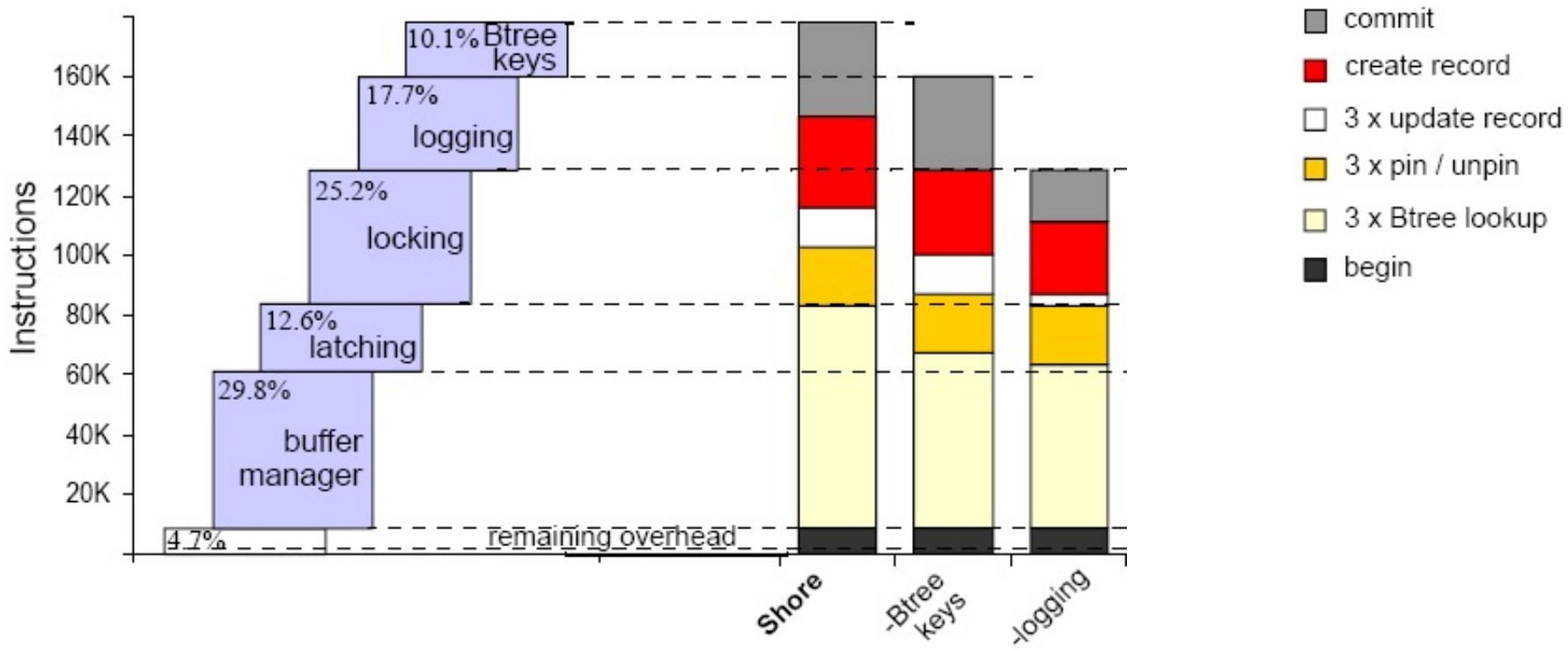
Instruction of useful work is only <2% of a memory resident DB



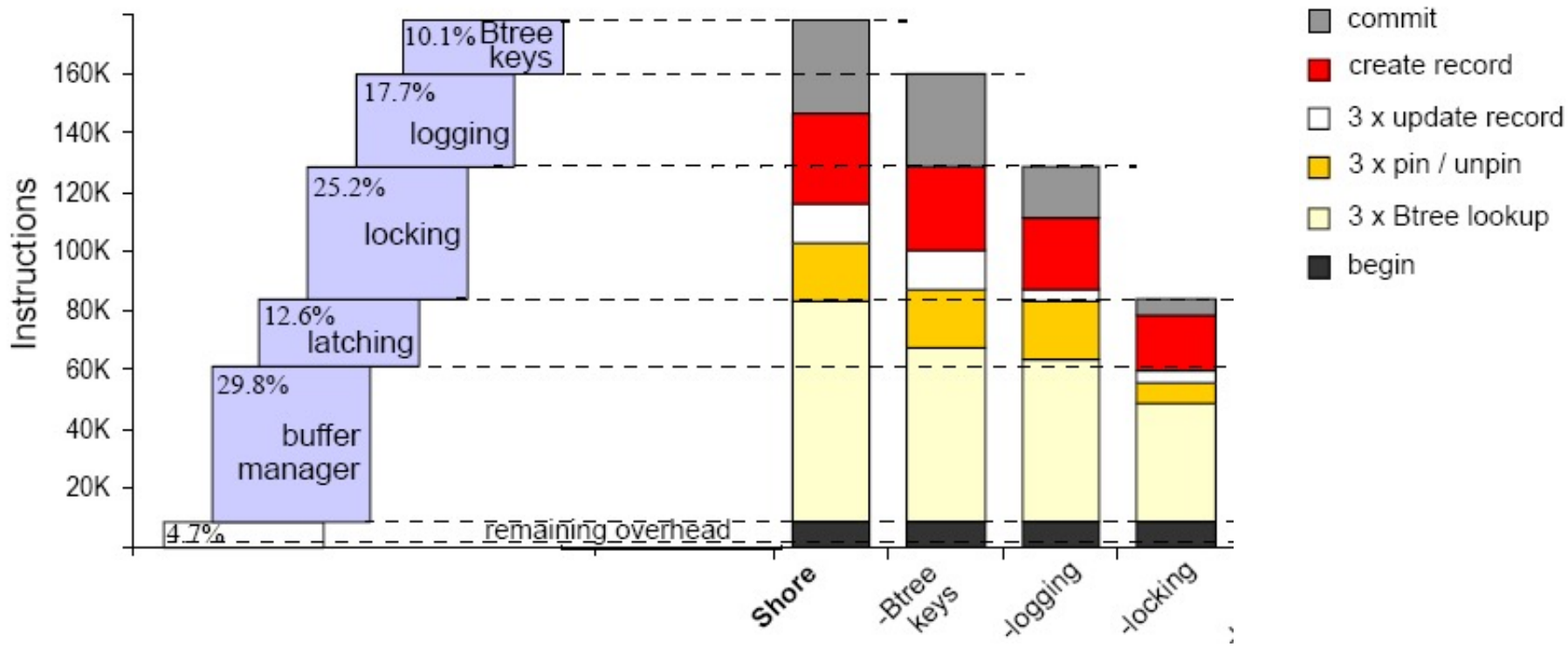
Effect of removing different components for payment (1/6)



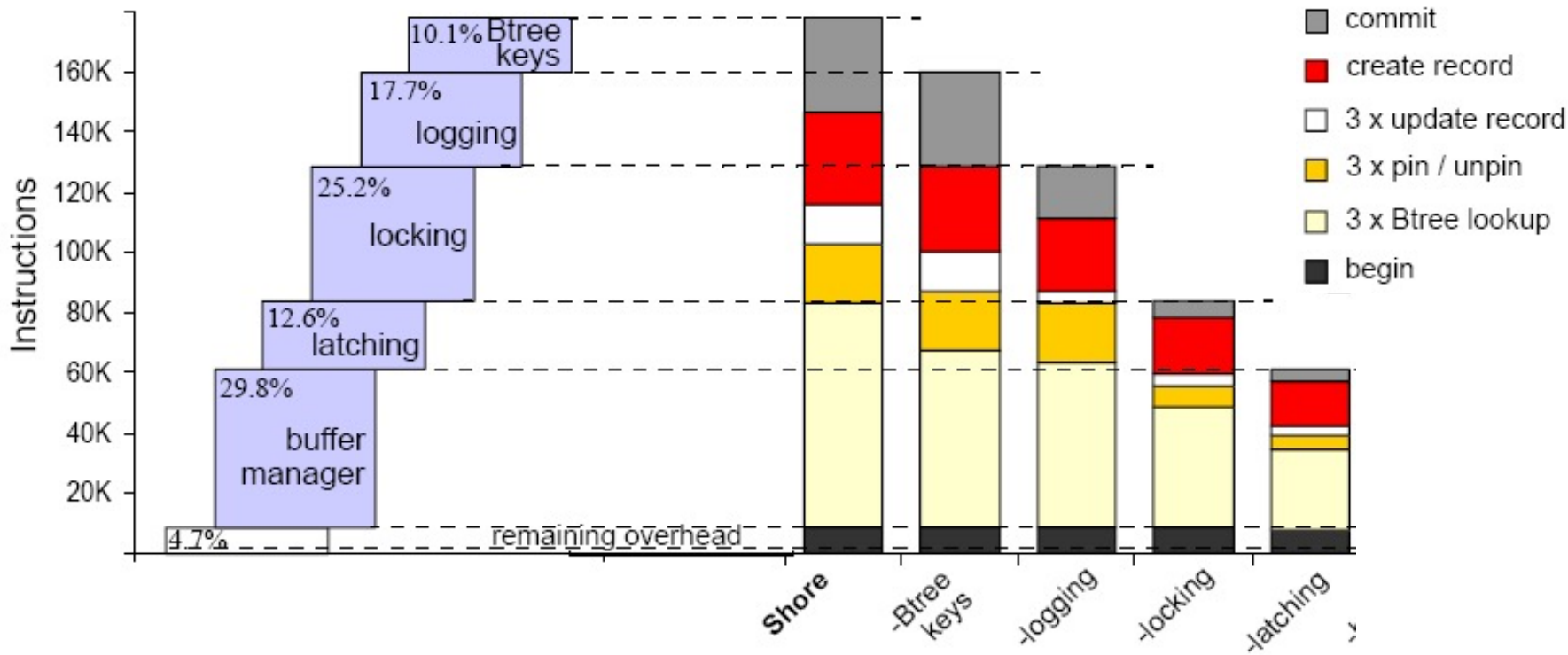
Effect of removing different components for payment (2/6)



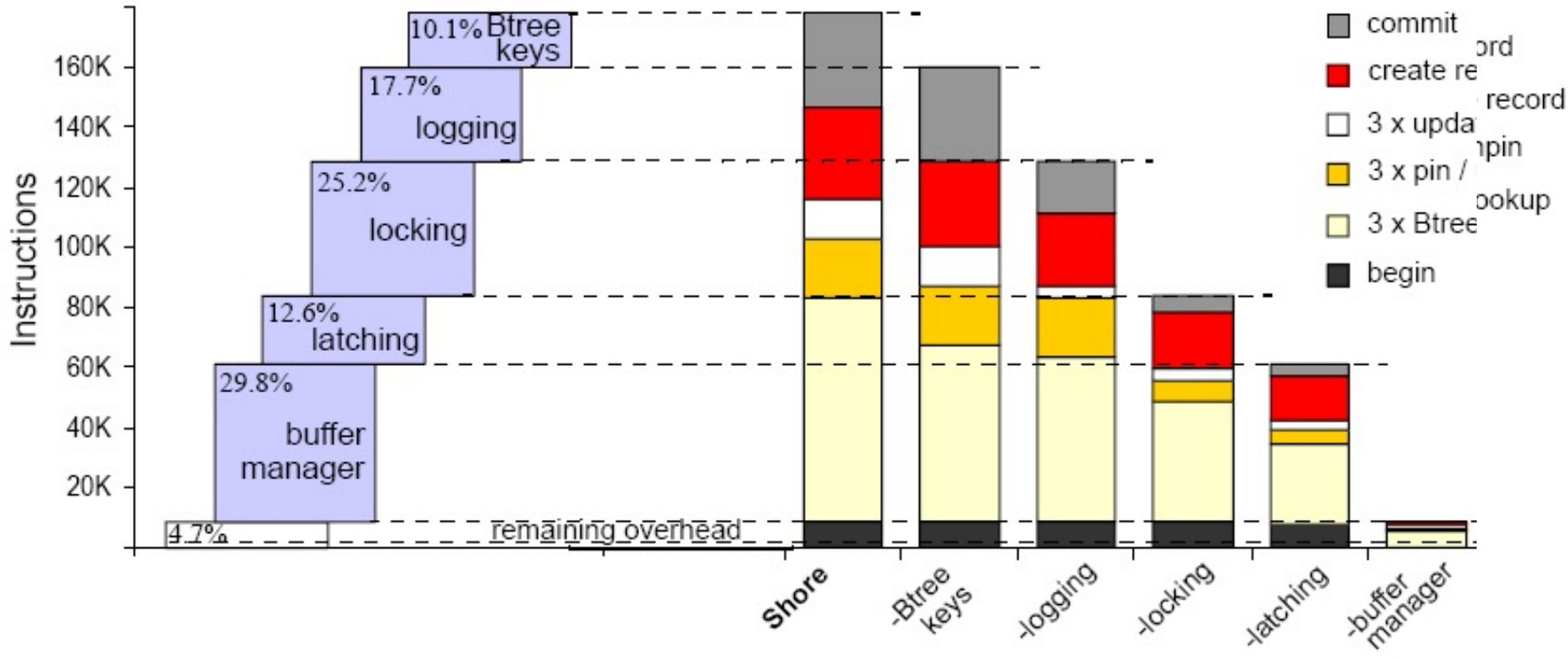
Effect of removing different components for payment (3/6)



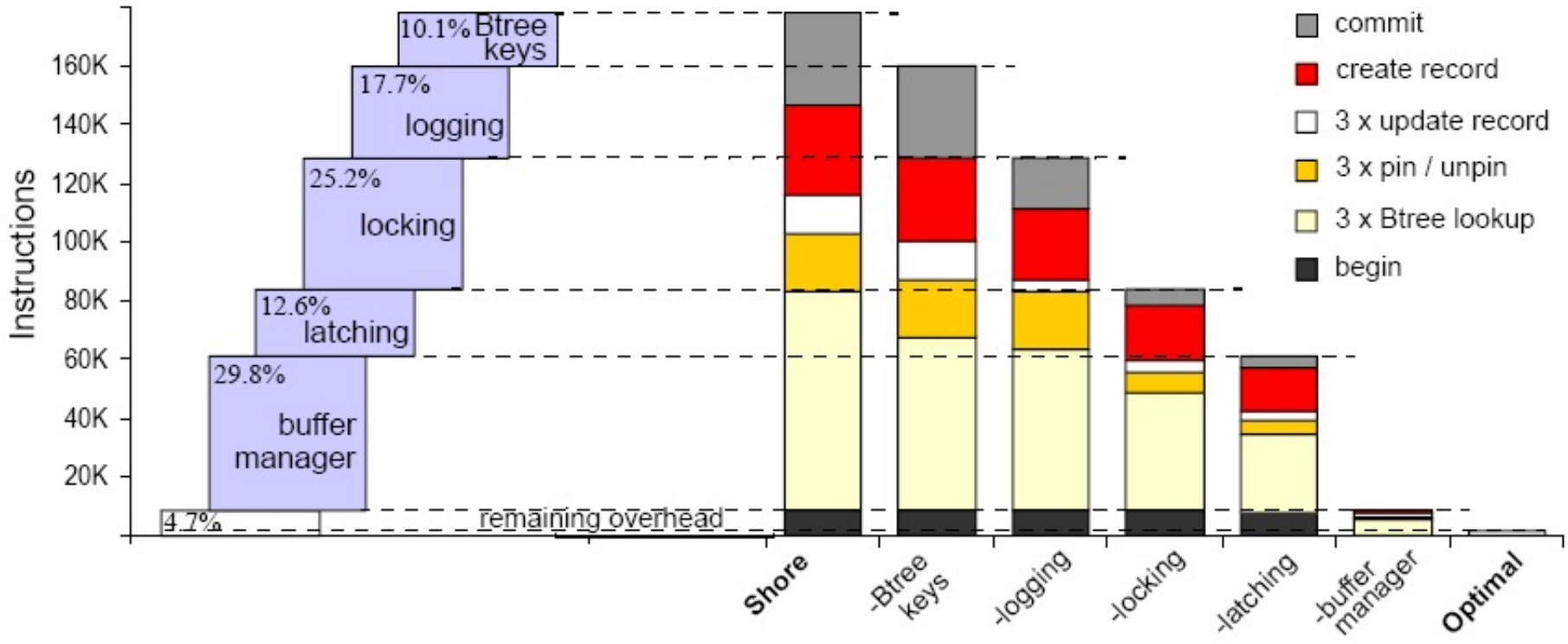
Effect of removing different components for payment (4/6)

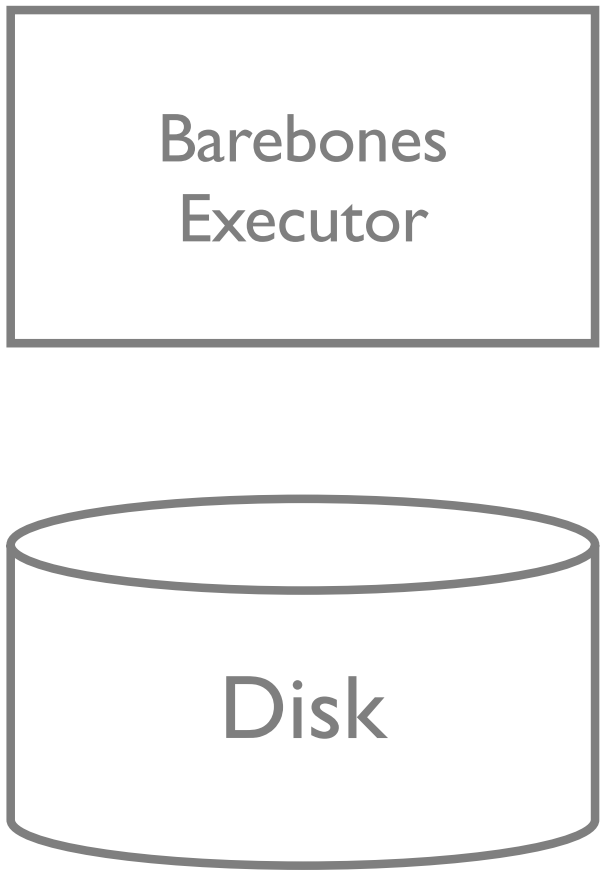


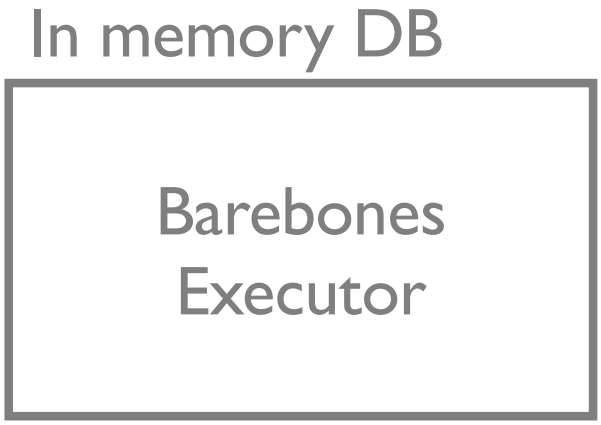
Effect of removing different components for payment (5/6)



Effect of removing different components for payment (6/6)

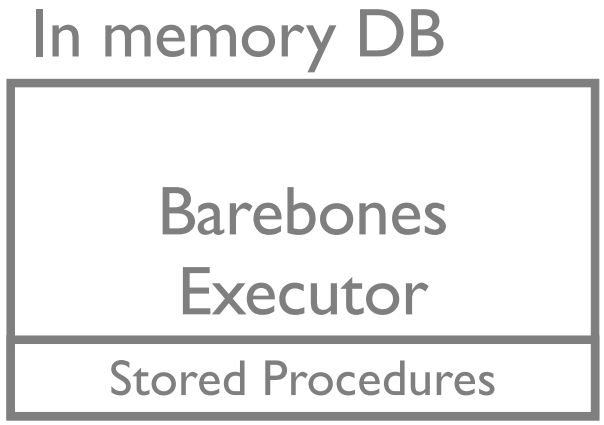



Barebones
Executor
Disk



What about

- Parsing
- Concurrency?
- Recovery?



What about

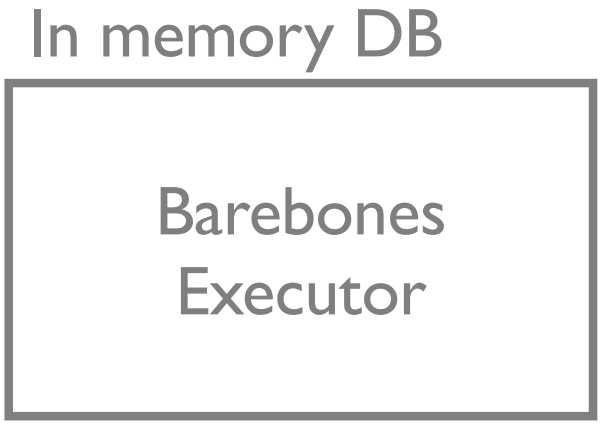
- Parsing
- Concurrency?
- Recovery?

Procedure:

```
p1 =  SELECT cost
      FROM fact_table
      WHERE id = ?
```

Query:

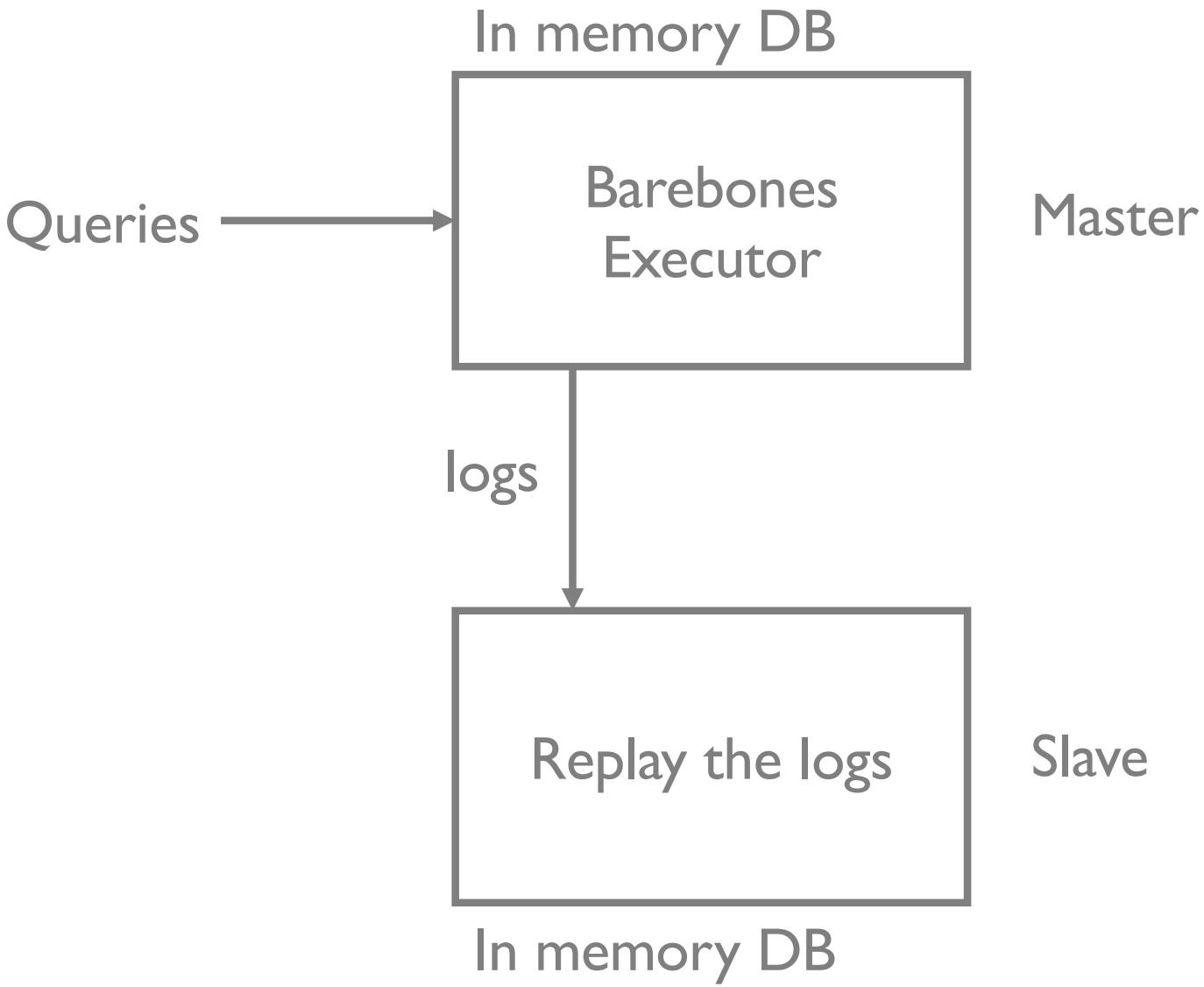
```
p1(10)
```



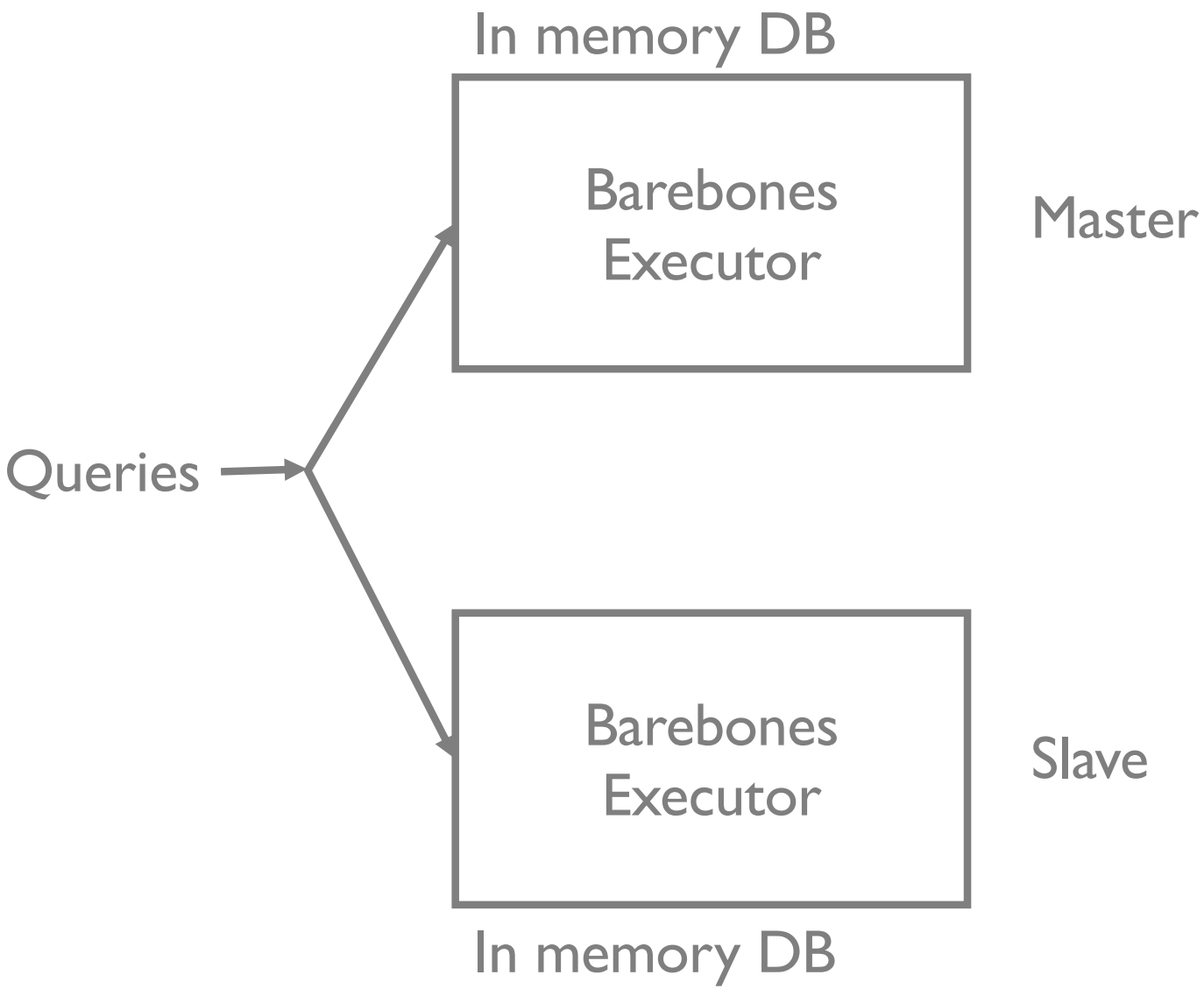
What about

- Parsing
- ~~Concurrency?~~
  - no buffer manager, no concurrency, no locks
- Recovery?

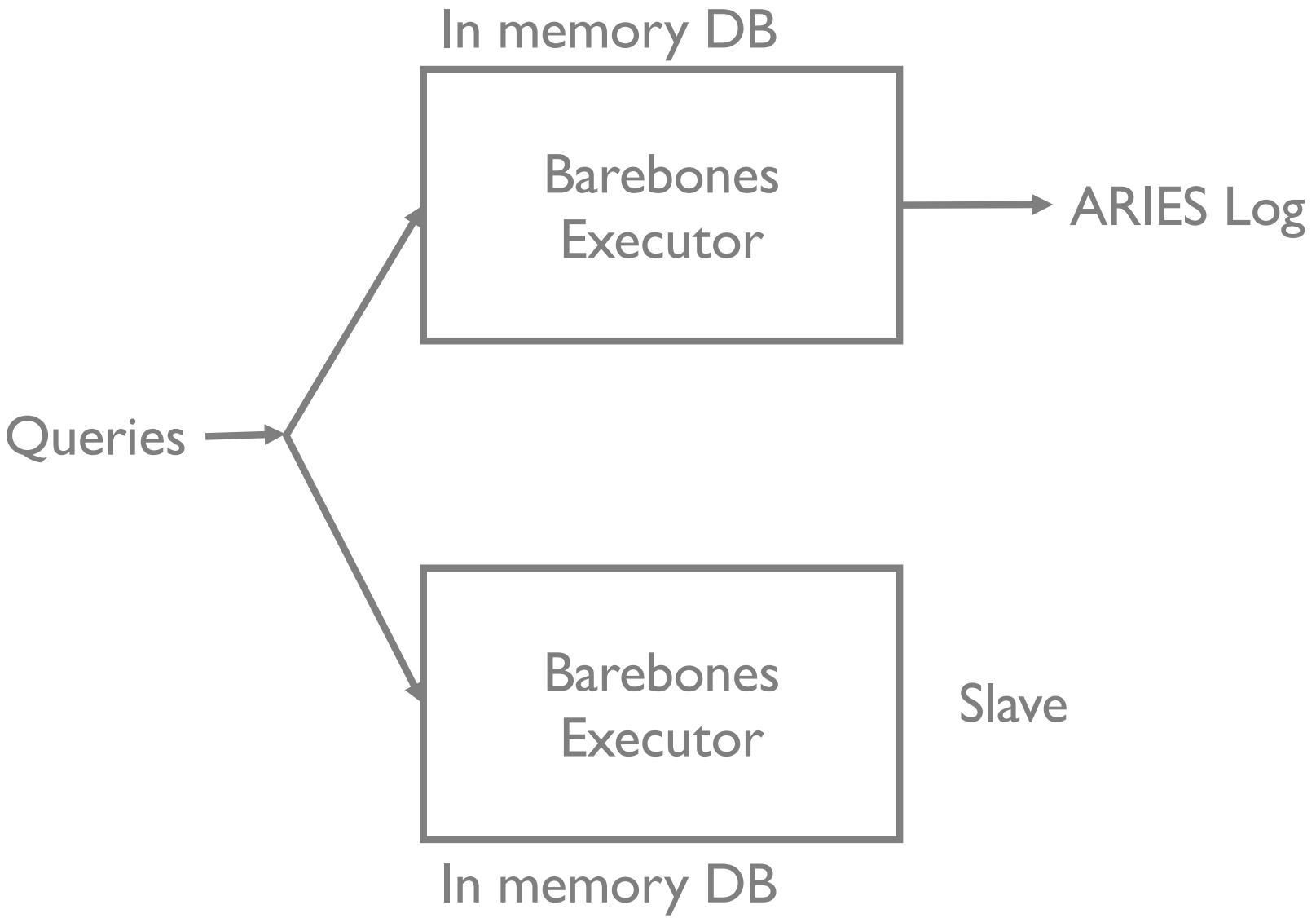
Log Shipping



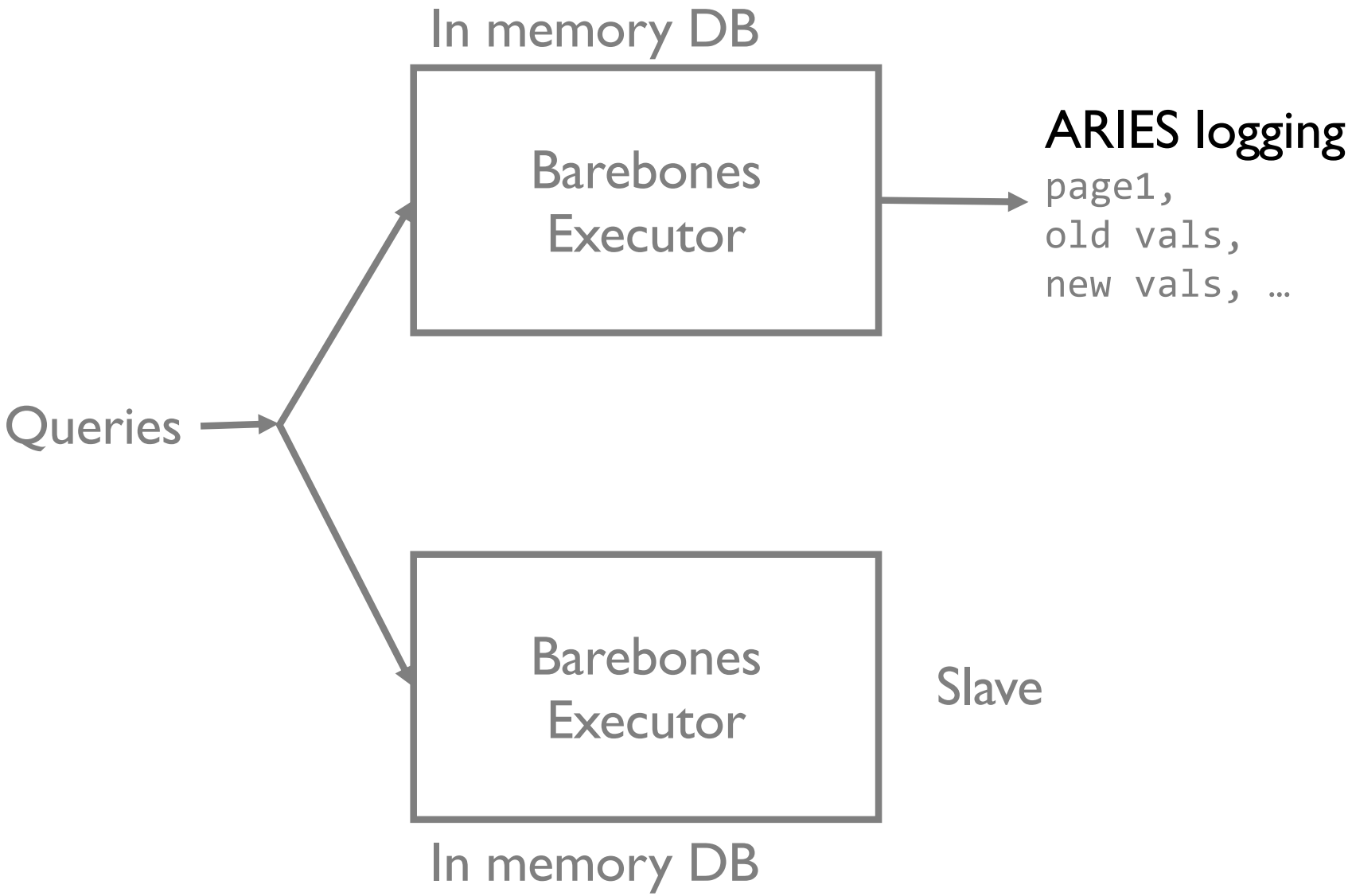
Active-Active



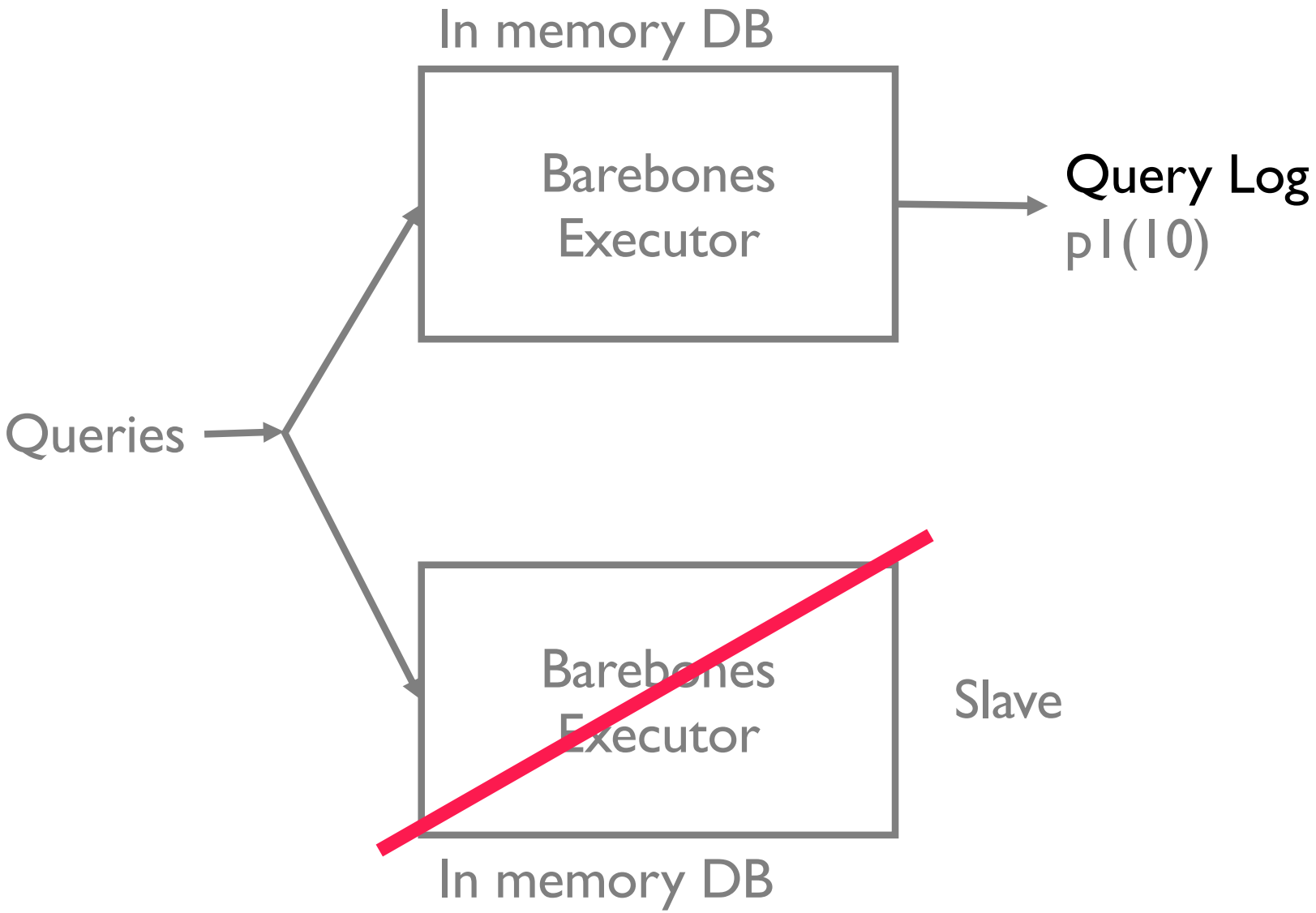
Recovery in Active-Active



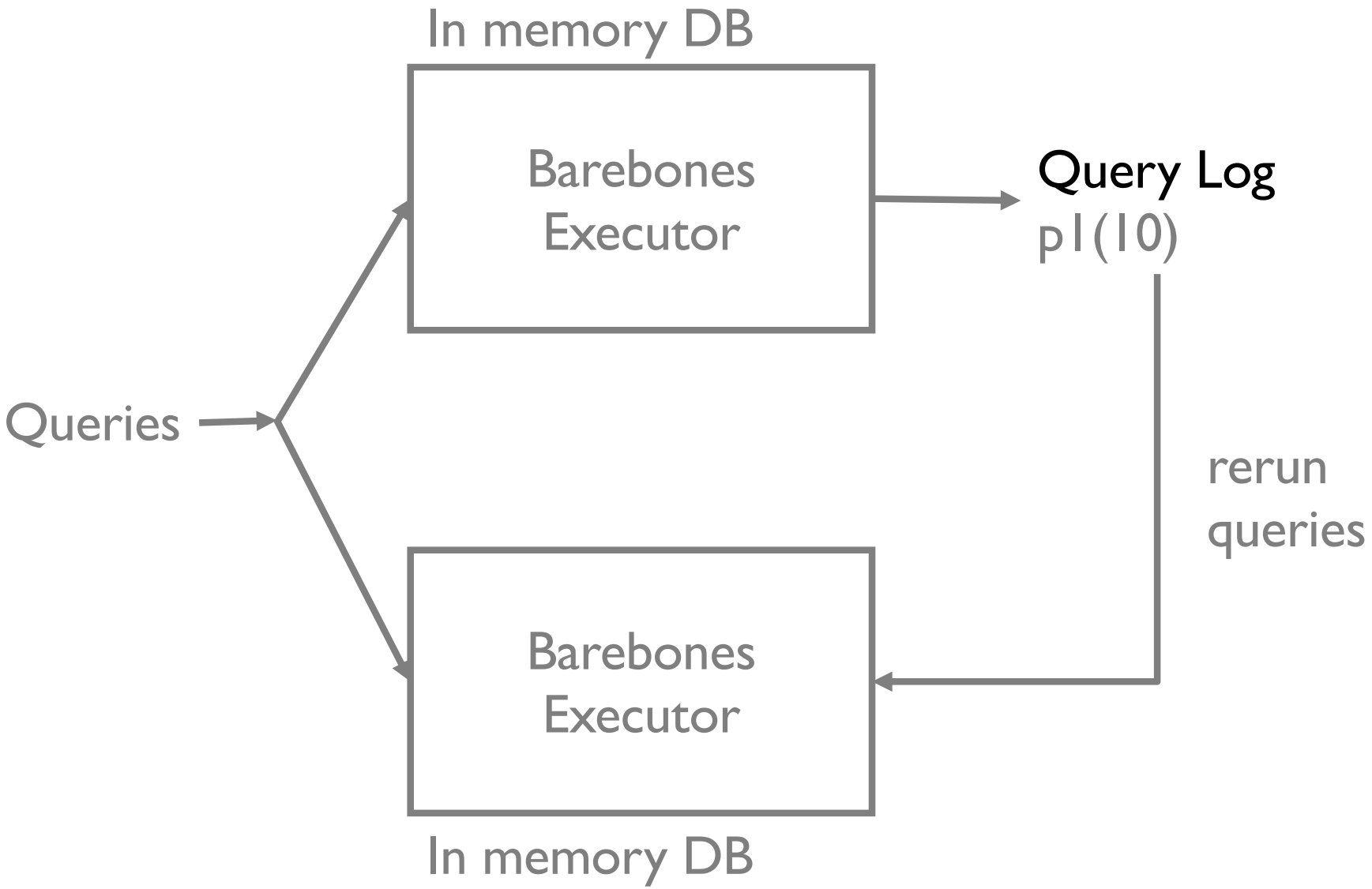
Recovery in Active-Active



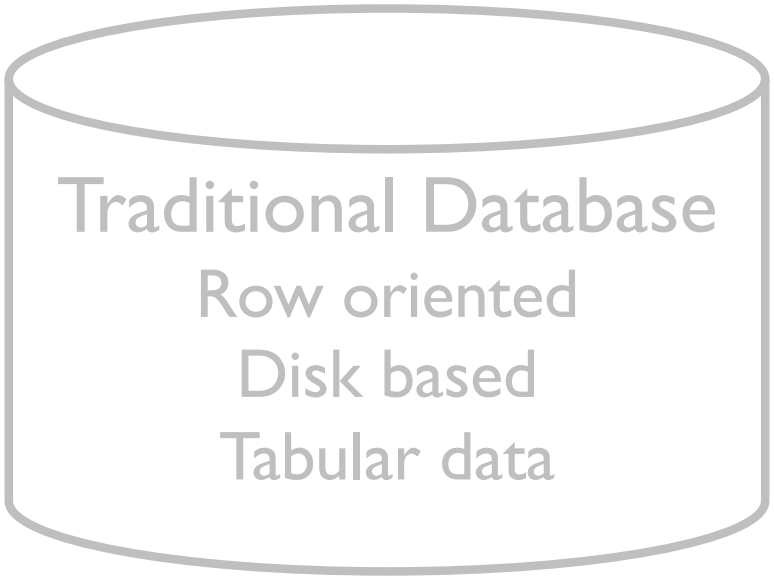
Recovery in Active-Active

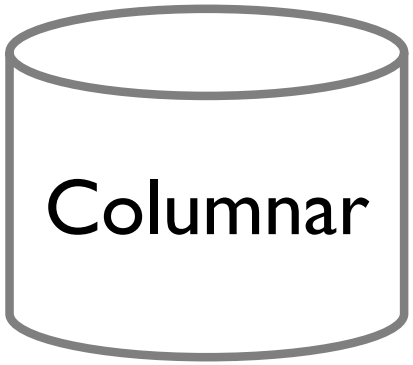


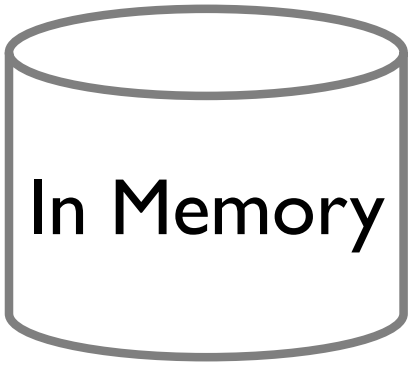
Recovery in Active-Active

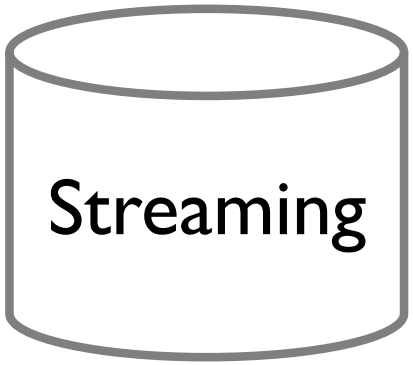


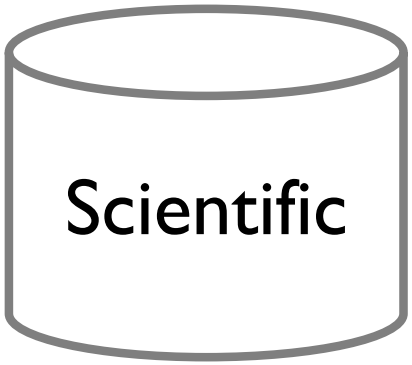
One Size Does Not Fits All



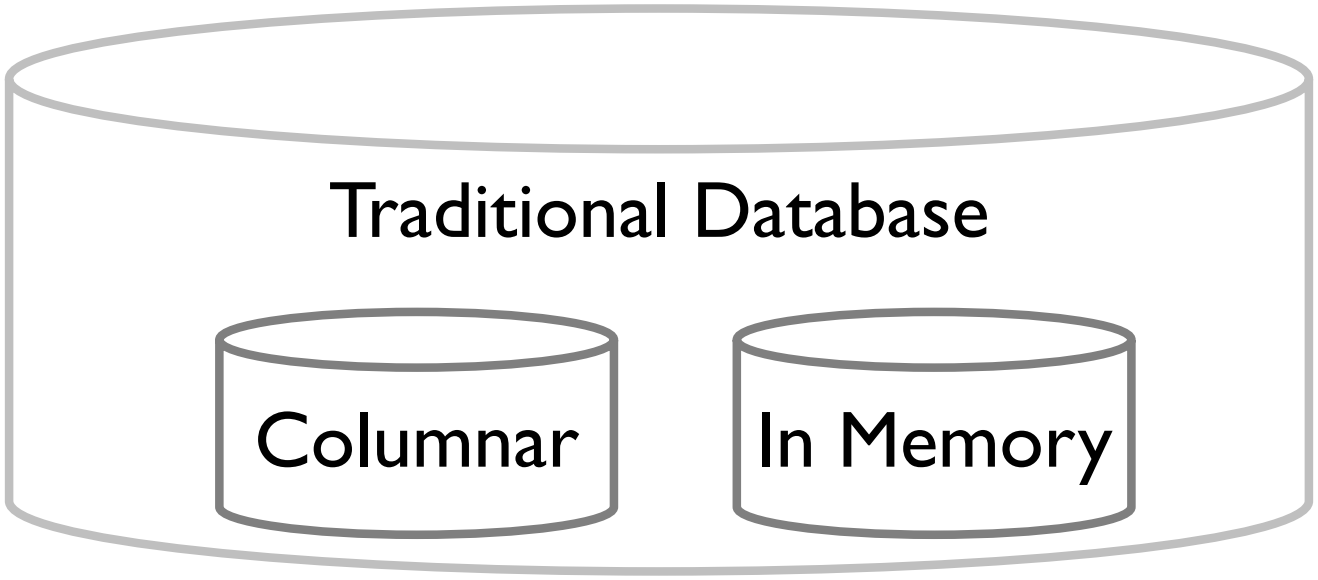




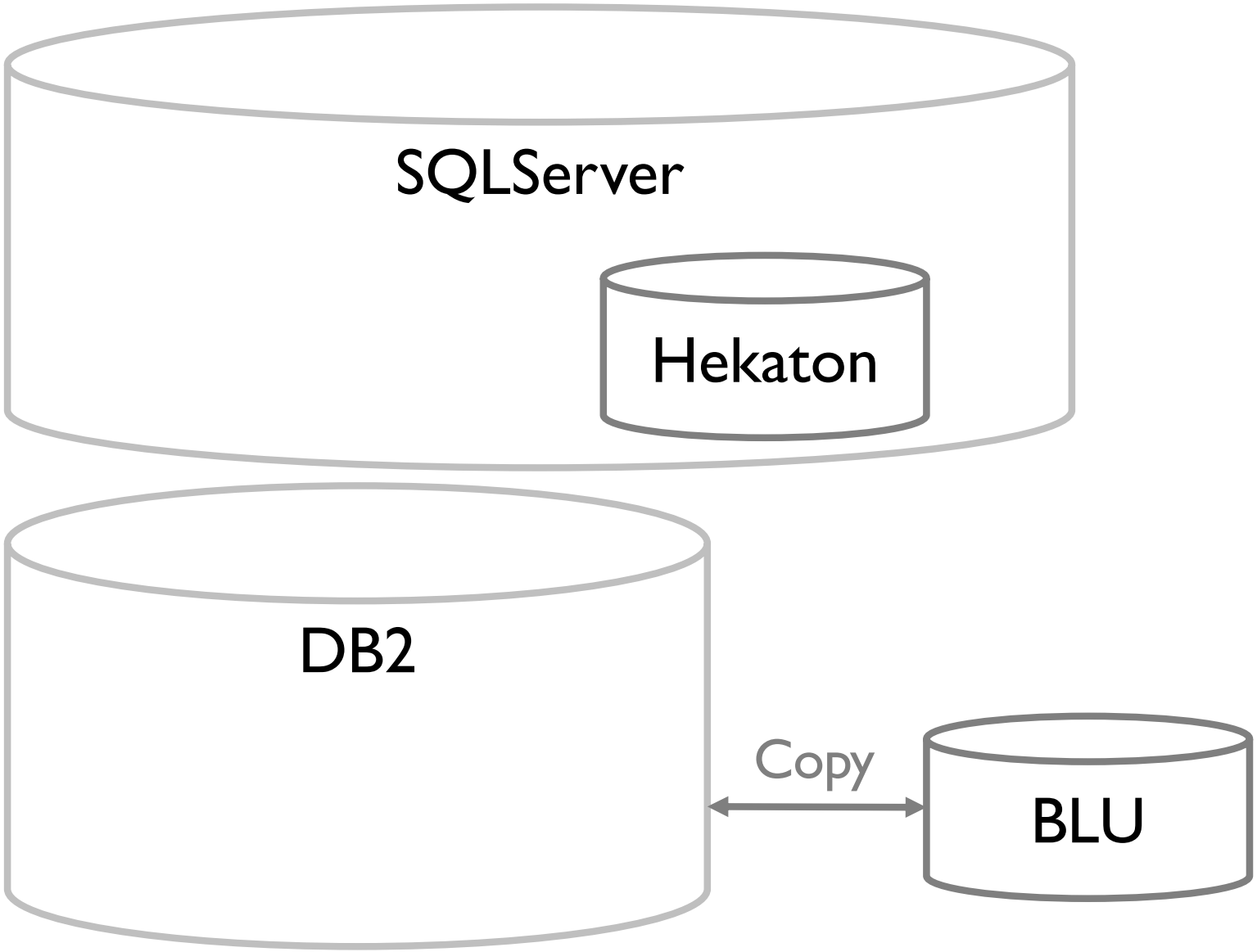




One Size Does Not Fits All



One Size Does Not Fits All

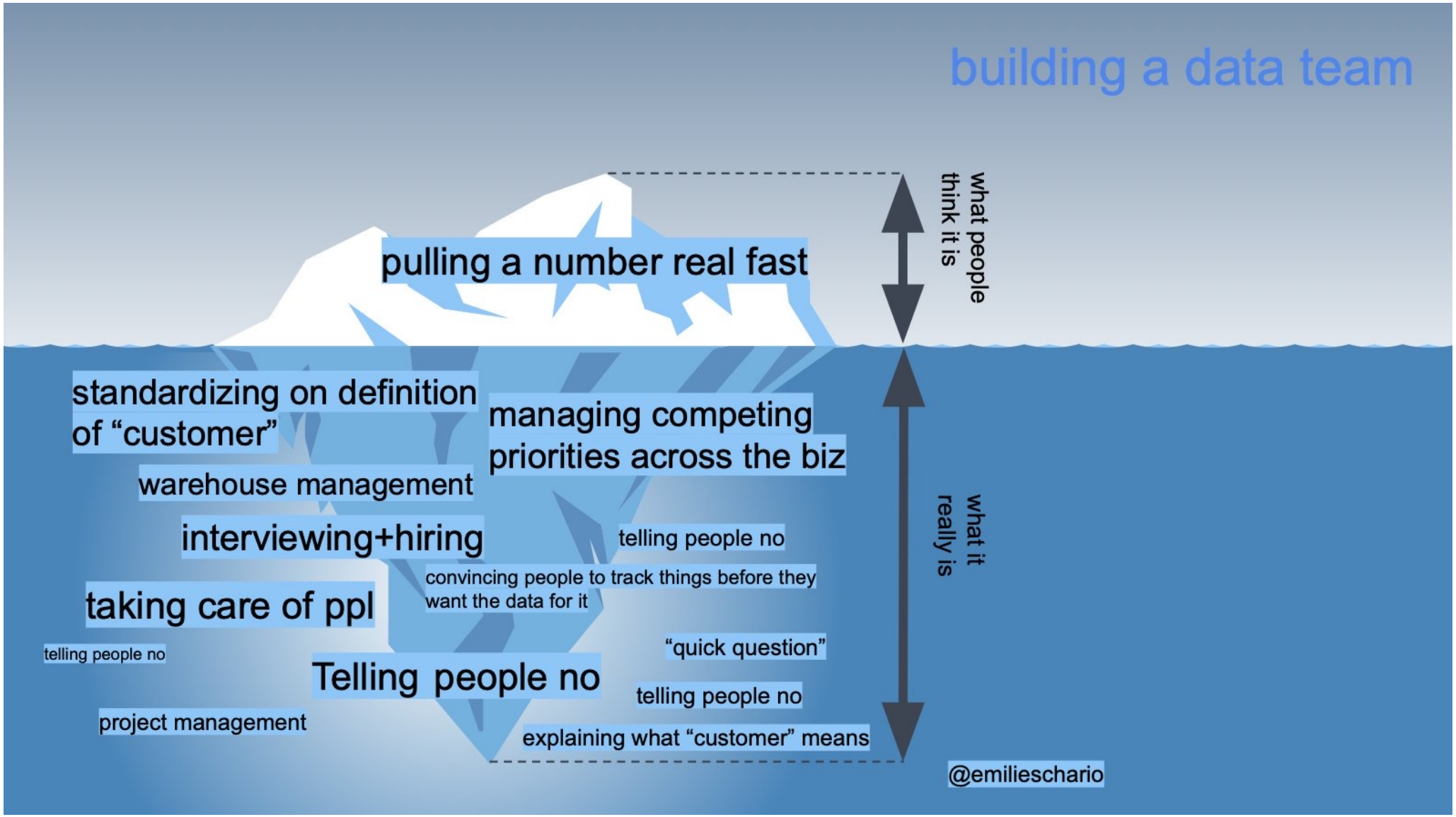


OK Let's Step Back

Discussed the *how*:

- how to model data needed by an application
- how to query databases
- how databases execute queries
- how databases run fast, correctly

To what end?



https://twitter.com/emilieschario/status/1427419697654419456/p

Production ML

An on-call engineer's biggest nightmare

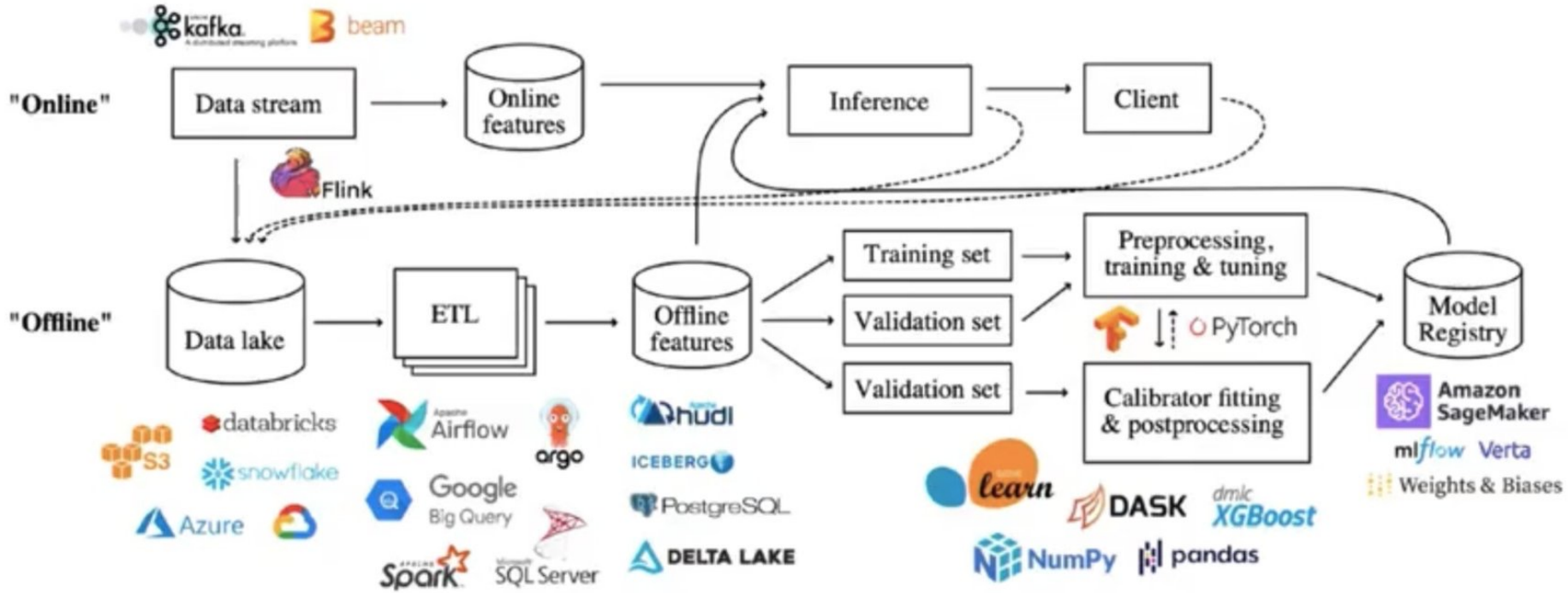


Figure 1: High-level architecture of a generic end-to-end machine learning pipeline. Logos represent a sample of tools used to construct components of the pipeline, illustrating heterogeneity in the tool stack. *Shankar et al. 2021*

https://www.facebook.com/Engineering/videos/1578607659138164/

4xxx: “here are facts about data management”

- 4111: basics
- 4112: gory DBMS internals

6xxx: “how people figured out those facts”

- 6111: info extraction and web
- 6113: class and modern DB research
- 6998: systems for human data interaction

PhD: “we don’t understand anything. Plz help”

Thanks!

Please fill out courseworks survey

cool.