COSI 167A
Advanced Data Systems

Class 3

Data Systems Architecture

Prof. Subhadeep Sarkar

Brandeis University

https://ssd-brandeis.github.io/COSI-167A/

Today in COSI 167A

What's on the cards?

fundamentals of data **storage**

introduction to **row-stores** and **column-stores**

Class **logistics**

and administrivia

Project 1 (C++/Java) has been released (due on **Sep 20**).

C/C++ learning resources at: https://ssd-brandeis.github.io/COSI-167A/assignments/

The **first technical question** is now available on the class website (due **before the class** on **Sep 10**).

Recap: **Storage hierarchy**

How data moves!

Volatile

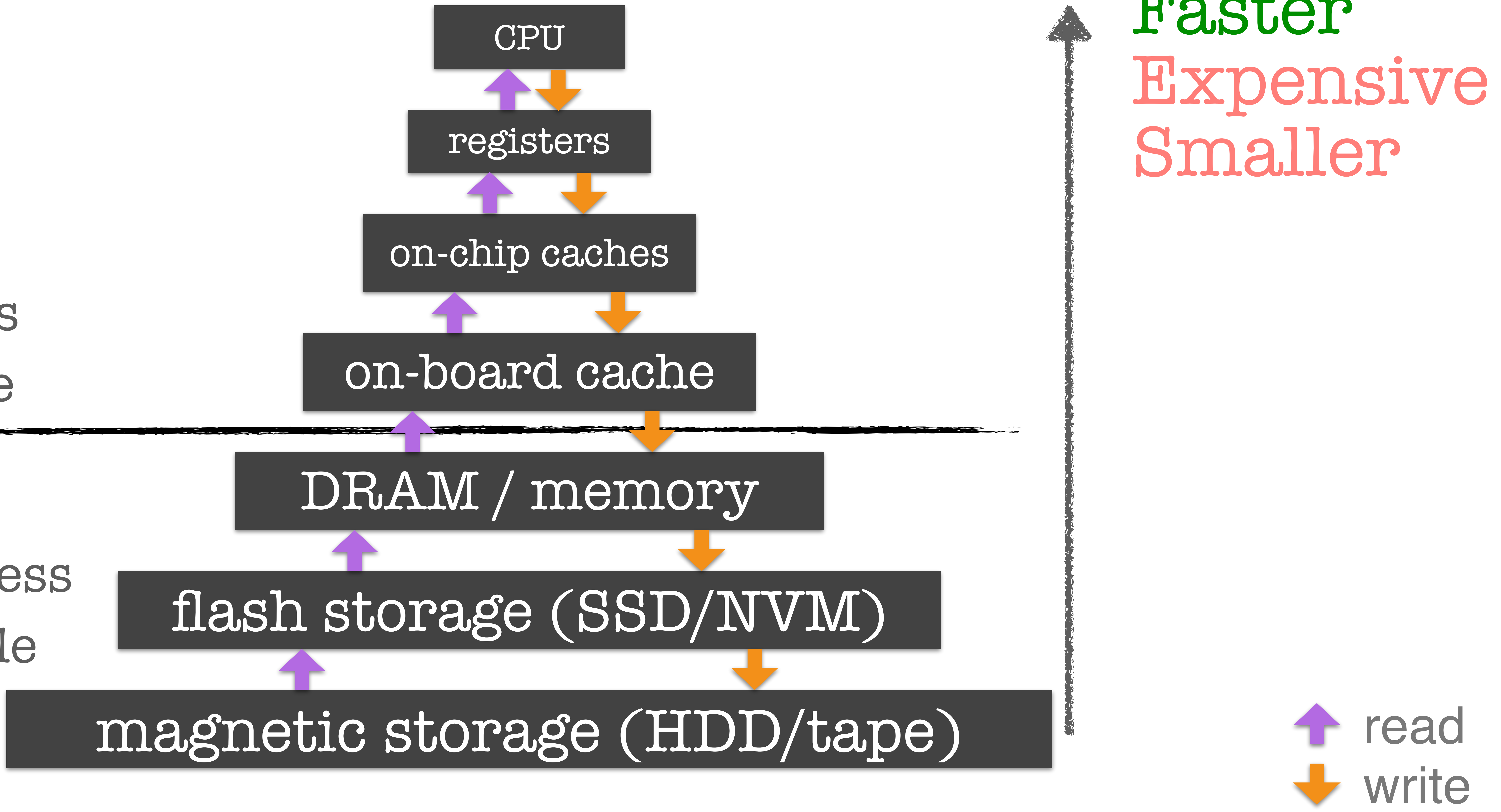
Random access

Byte accessible

Non-volatile

Sequential access

Block accessible



Brandeis UNIVERSITY

Memory wall

Try not to jump the wall

computations happen here

CPU

1 ns register

4 ns on-chip cache

10 ns on-board cache

100 ns memory

flash storage

magnetic storage

be careful when you go below the green line

Brandeis UNIVERSITY

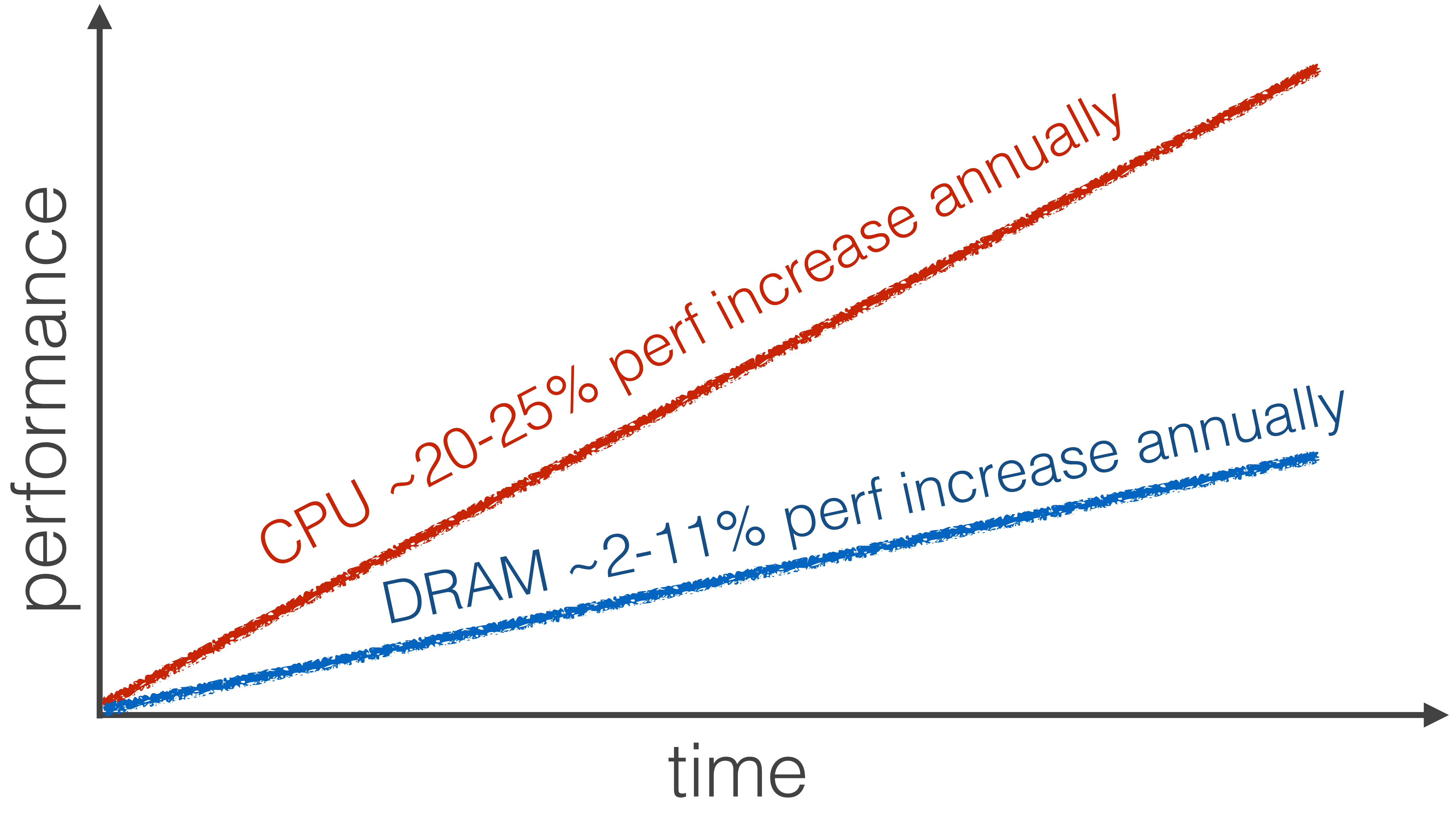
Memory wall

Try not to jump the wall

computations happen here

	CPU
1 ns	register
4 ns	on-chip cache
10 ns	on-board cache
100 ns	memory
	flash storage
	magnetic storage

be careful when you go below the green line



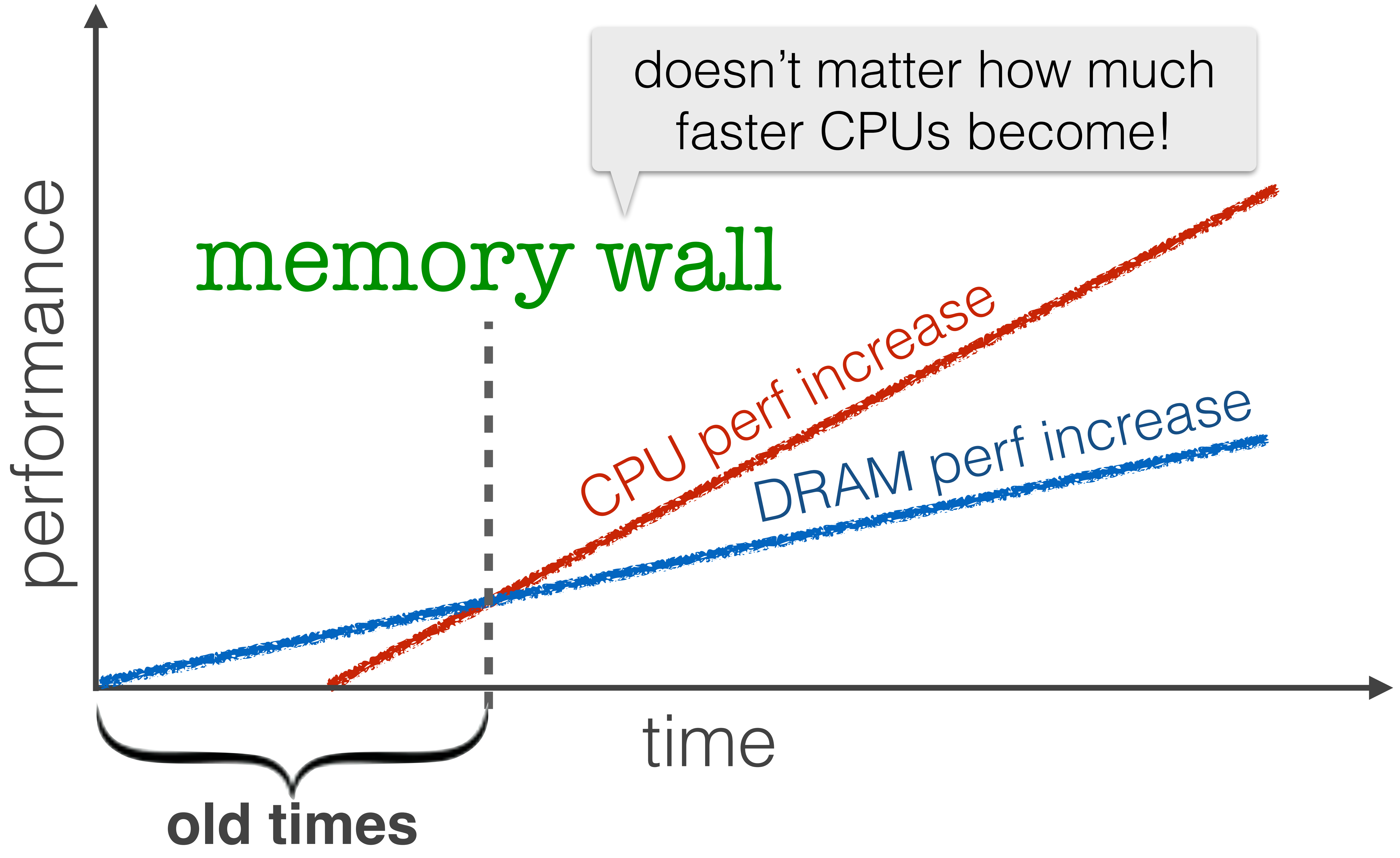
Memory wall

Try not to jump the wall

computations happen here

	CPU
1 ns	register
4 ns	on-chip cache
10 ns	on-board cache
100 ns	memory
	flash storage
	magnetic storage

be careful when you go below the green line



Brandeis UNIVERSITY

Memory wall

Try not to jump the wall

computations happen here

CPU

1 ns	register
4 ns	on-chip cache
10 ns	on-board cache
100 ns	memory
16,000 ns	flash storage
2 ms	magnetic storage

be careful when you go below the green line

be VERY careful when you go below the red line

Recap: **Storing** data

Things to keep in mind

Disk is 6 orders of magnitude slower than CPU

SSDs are 4 orders of magnitude slower

Memory is 3 orders of magnitude slower

Recap: **Random** vs. **Sequential** access

So, be VERY careful!

Avoid disk accesses (reads/writes) whenever possible

I/Os to secondary storage is *always* **slow**!

Sequential access

read **each block exactly once**; process it; discard it; read next block

modern hardware can predict and **prefetch**; maximize performance

Random access

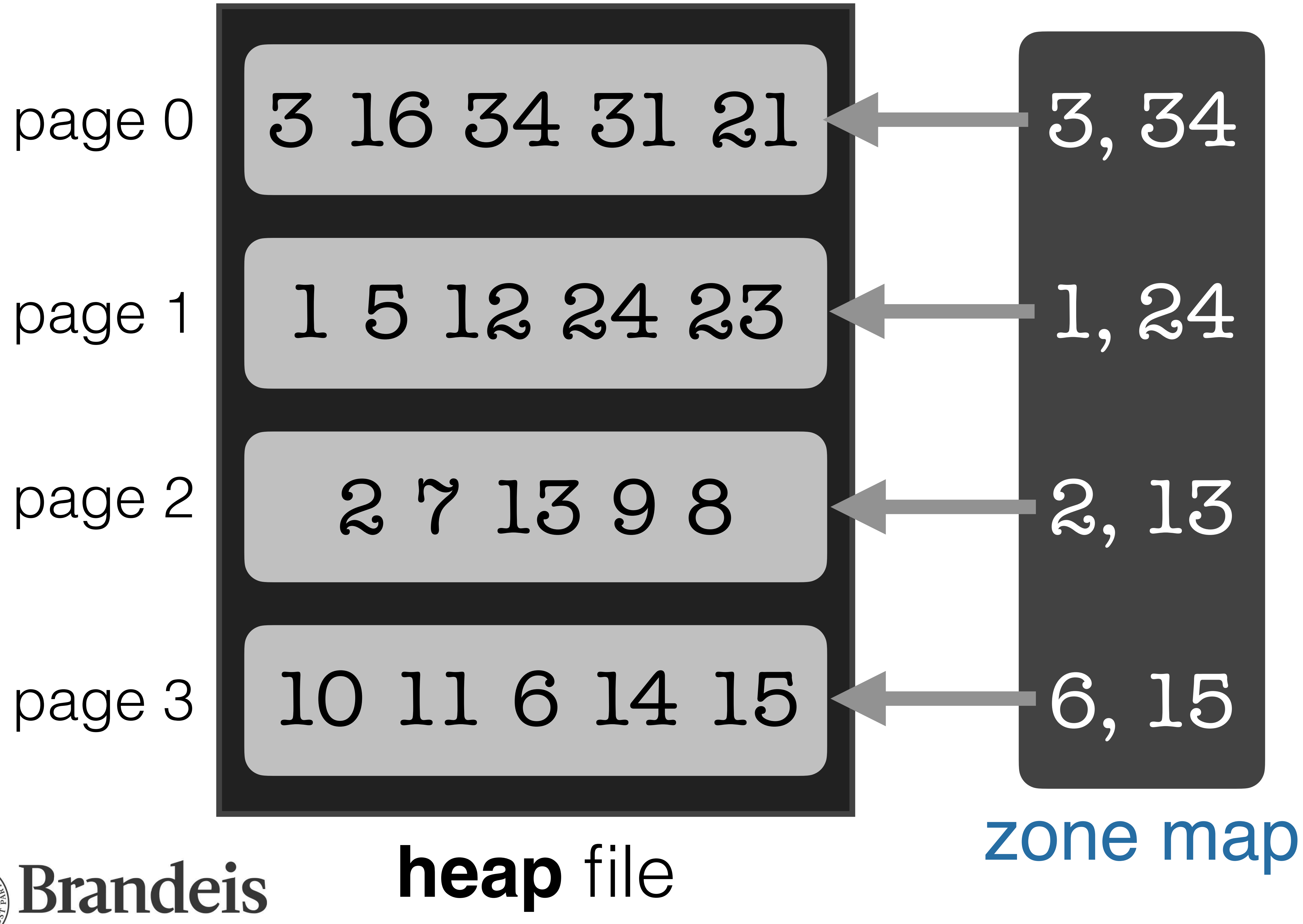
read a block; process it **partially**; discard it; may **read** the block **again**

often leads to **read amplification**

Project 1

Testing the waters!

Implementing a Simple Zone Map



w/o ZM: queries take 4 I/Os

with ZM: query: x<4: 3 I/Os

x<12: 4 I/O

x=1: 1 I/O

x=20: 4 I/O

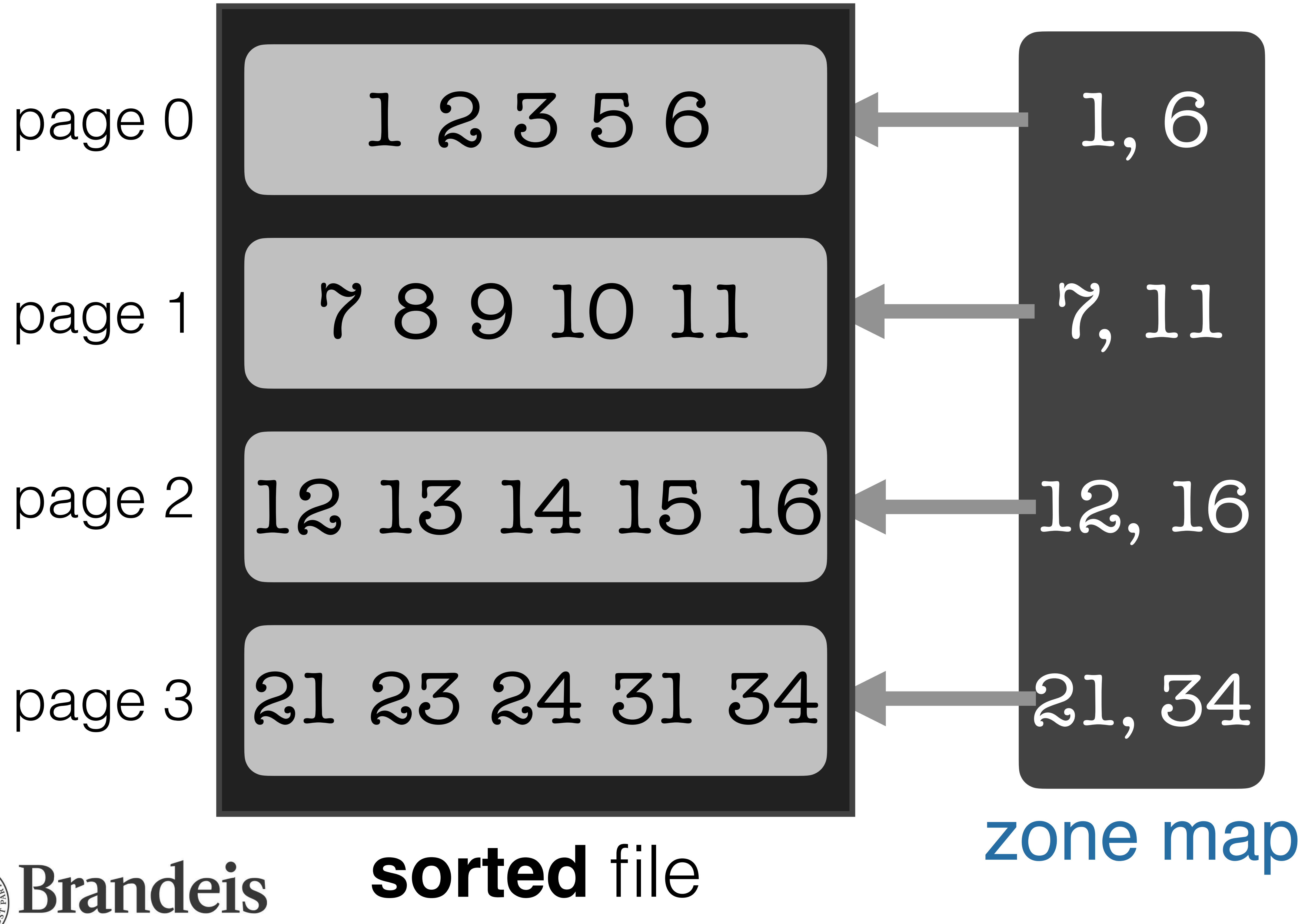
Thought Experiment

Are **zone maps** more or less useful if data is **sorted**?

Project 1

Testing the waters!

Implementing a Simple Zone Map



w/o ZM: queries take 4 I/Os

with ZM: query: x<4: 1 I/O

x<12: 2 I/Os

x=1: 1 I/O

x=20: 0 I/Os

Sorting is inherent indexing!

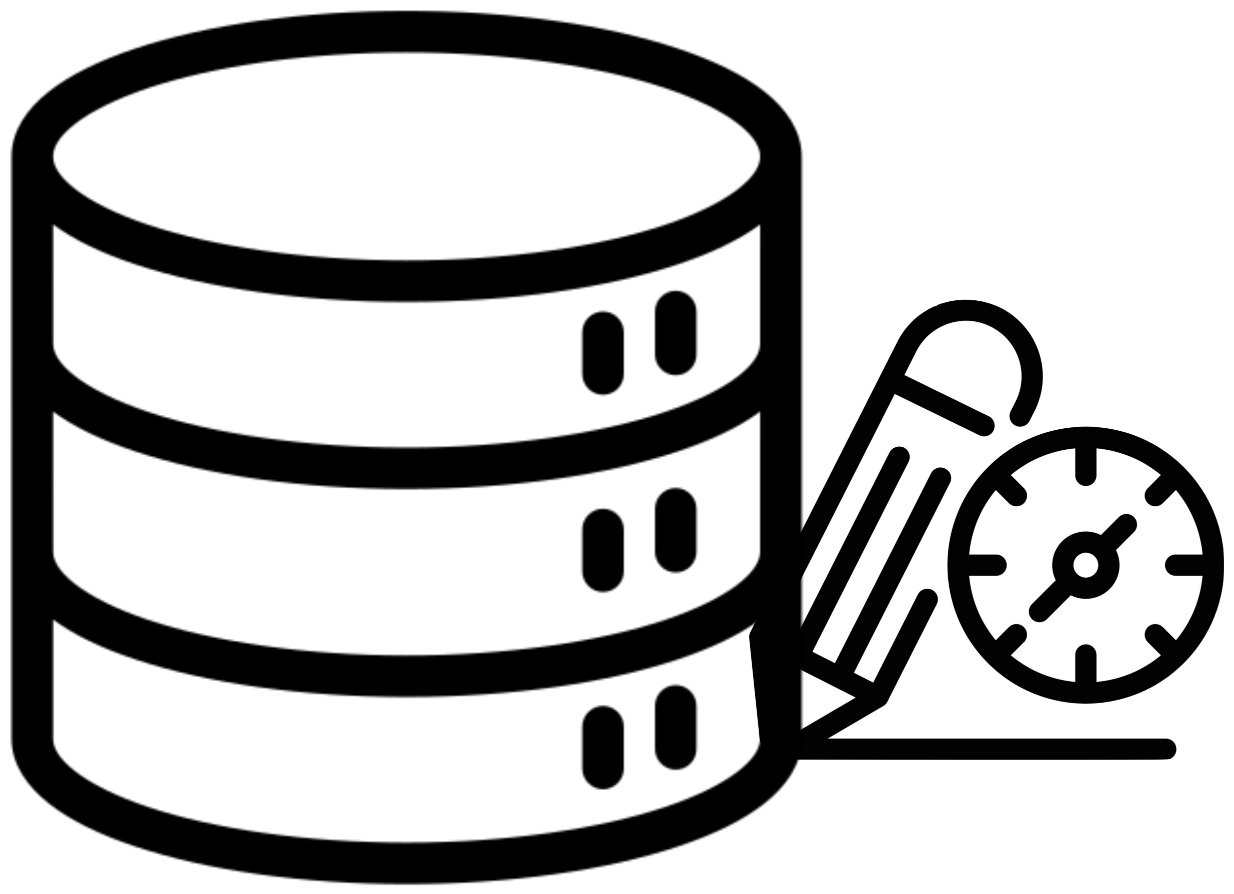
Zone map is a second index!

Brandeis UNIVERSITY

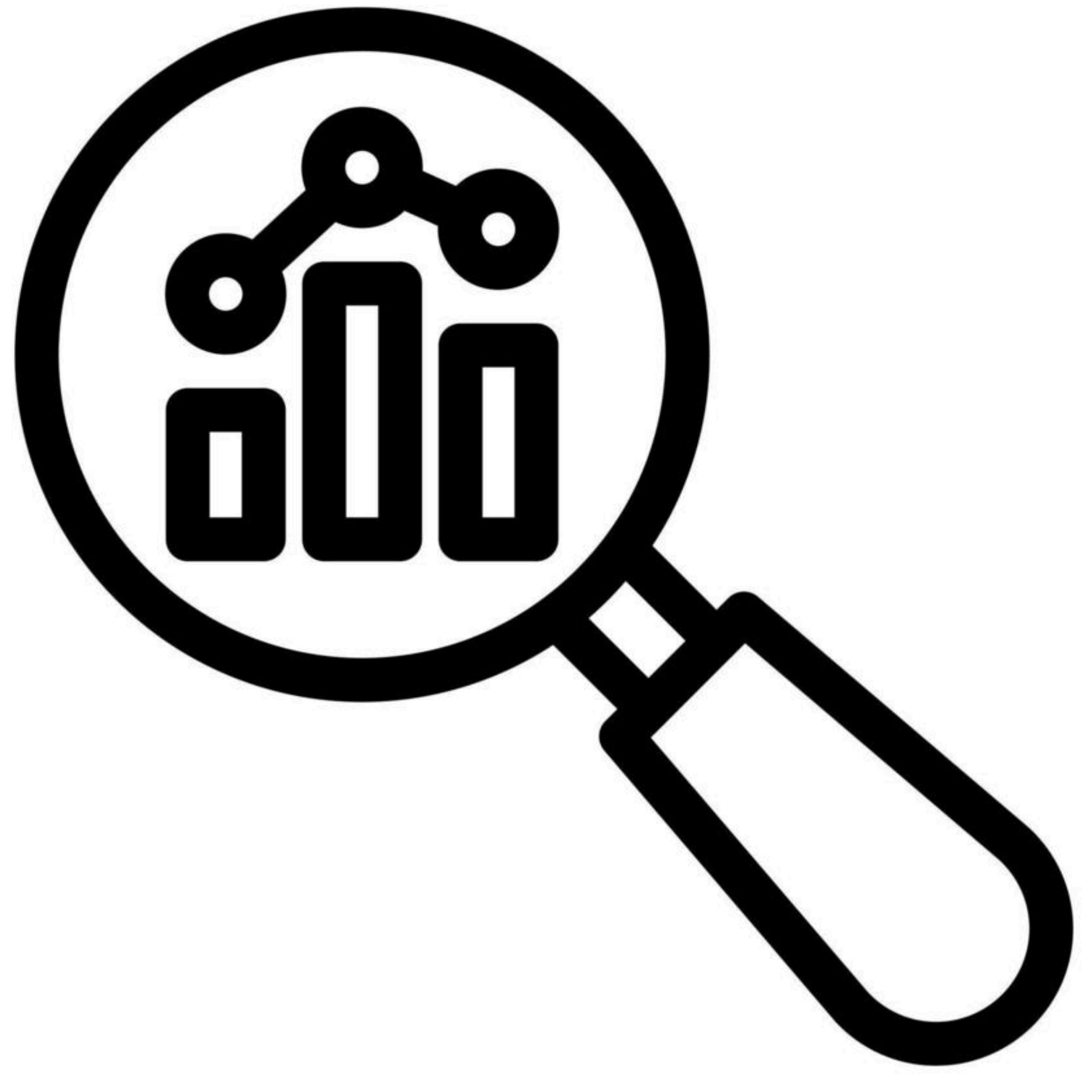
How we store (write) data heavily determines the performance of the system

The **design goals**

Biulding “efficient” data systems



build databases that can **write** data **fast** …



… and **process/analyze** that data **quickly**

Well, just **get to work** then!

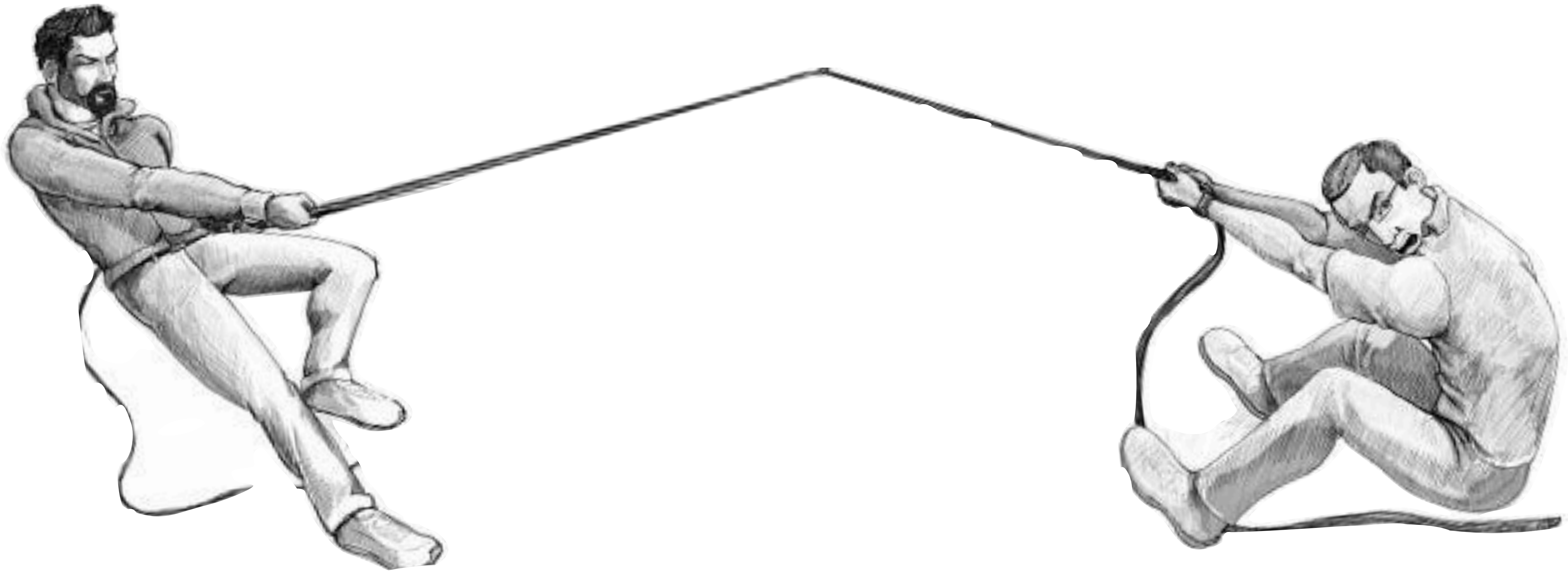
Performance **tradeoff**

The tug of war

hashmap

R: $\mathcal{O}(1)$

U: $\mathcal{O}(1)$



sorted array

R: $\mathcal{O}(log N)$

U: $\mathcal{O}(N)$

Query / **R**ead

Insert / **U**pdate

log

R: $\mathcal{O}(N)$

U: $\mathcal{O}(1)$

Performance **tradeoff**

The tug of war

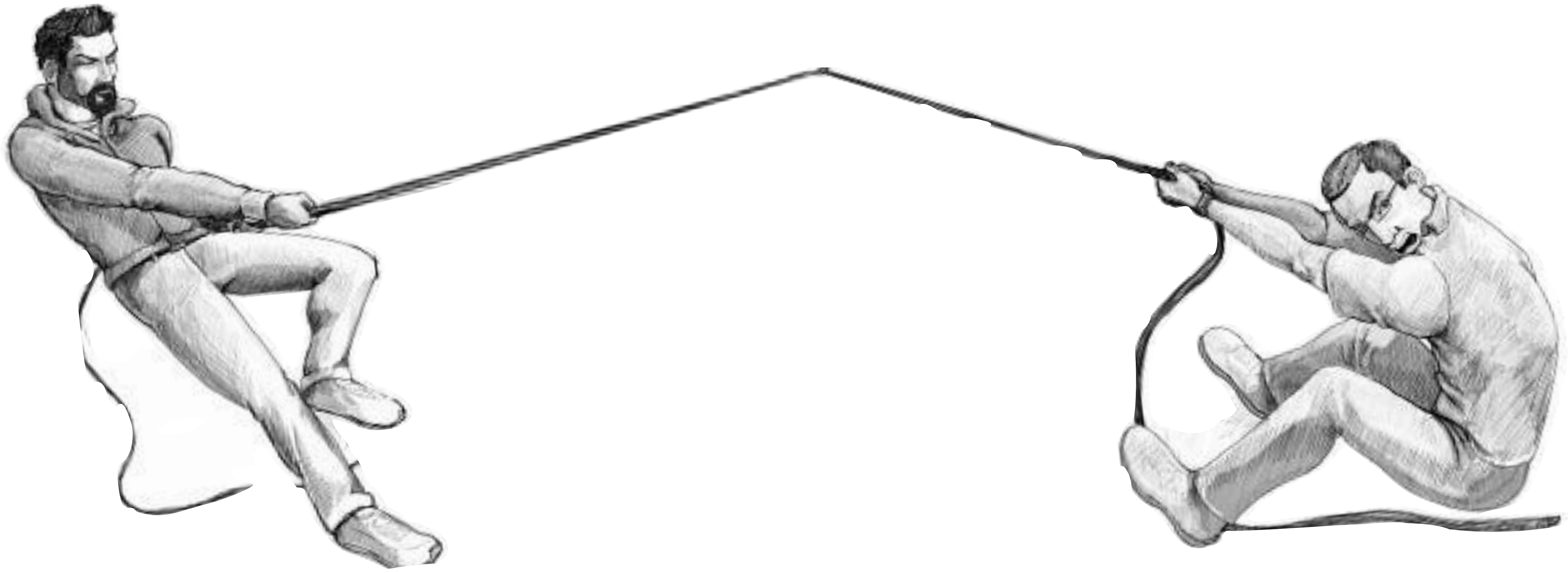
Storage / **M**emory

hashmap

M

R: $\mathcal{O}(1)$

U: $\mathcal{O}(1)$



sorted array

M

R: $\mathcal{O}(log N)$

U: $\mathcal{O}(N)$

Query / **R**ead

Insert / **U**pdate

log

M

R: $\mathcal{O}(N)$

U: $\mathcal{O}(1)$

Performance **tradeoff**

The tug of war

Storage / **M**emory

hashmap

R: $\mathcal{O}(1)$

M

There is **NO** perfect data structure!

sorted array

R: $\mathcal{O}(logN)$

U: $\mathcal{O}(N)$

M

Query / **R**ead

Insert / **U**pdate

log

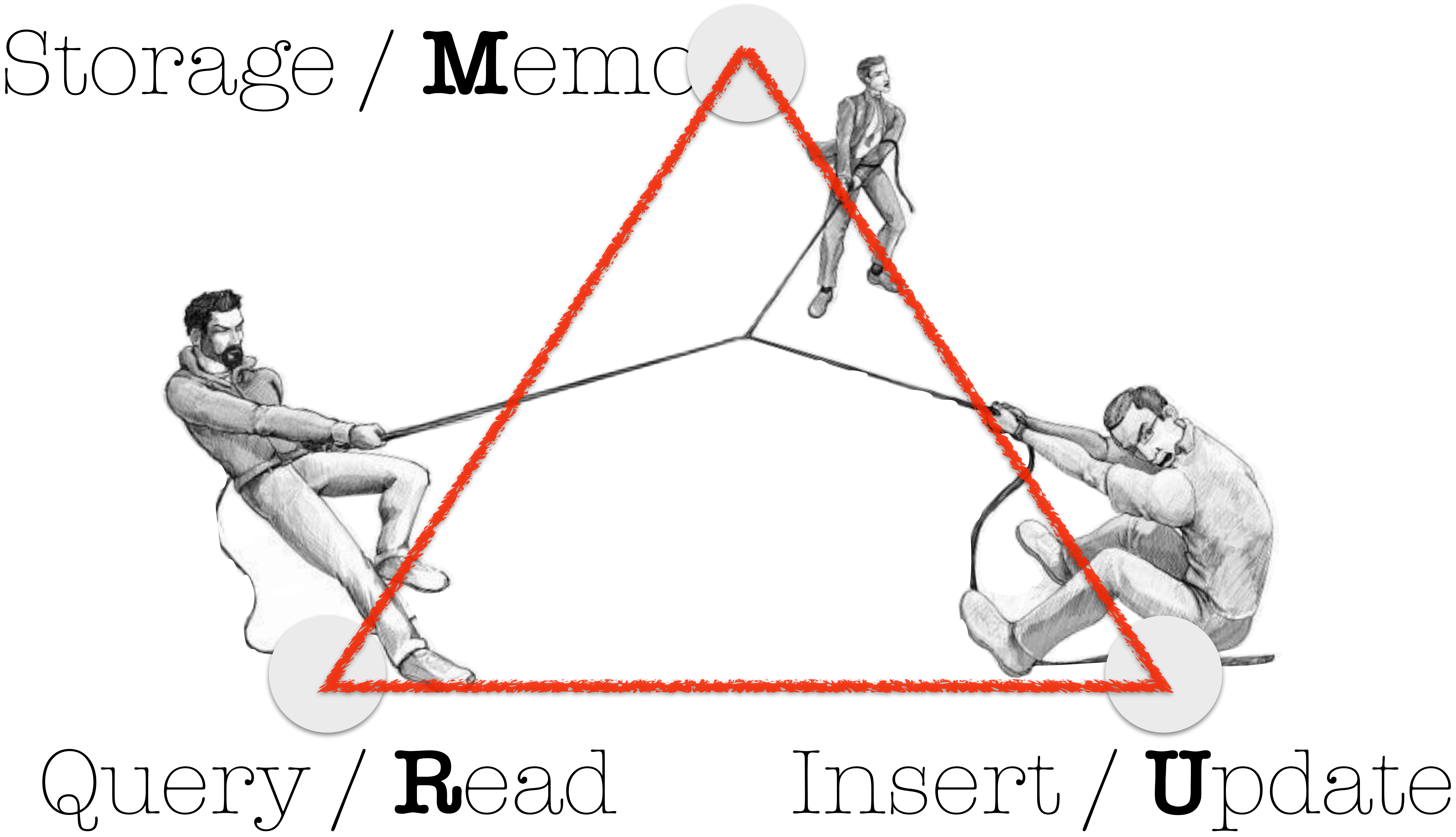
R: $\mathcal{O}(N)$

U: $\mathcal{O}(1)$

M

RUM conjecture

A three-way tradeoff

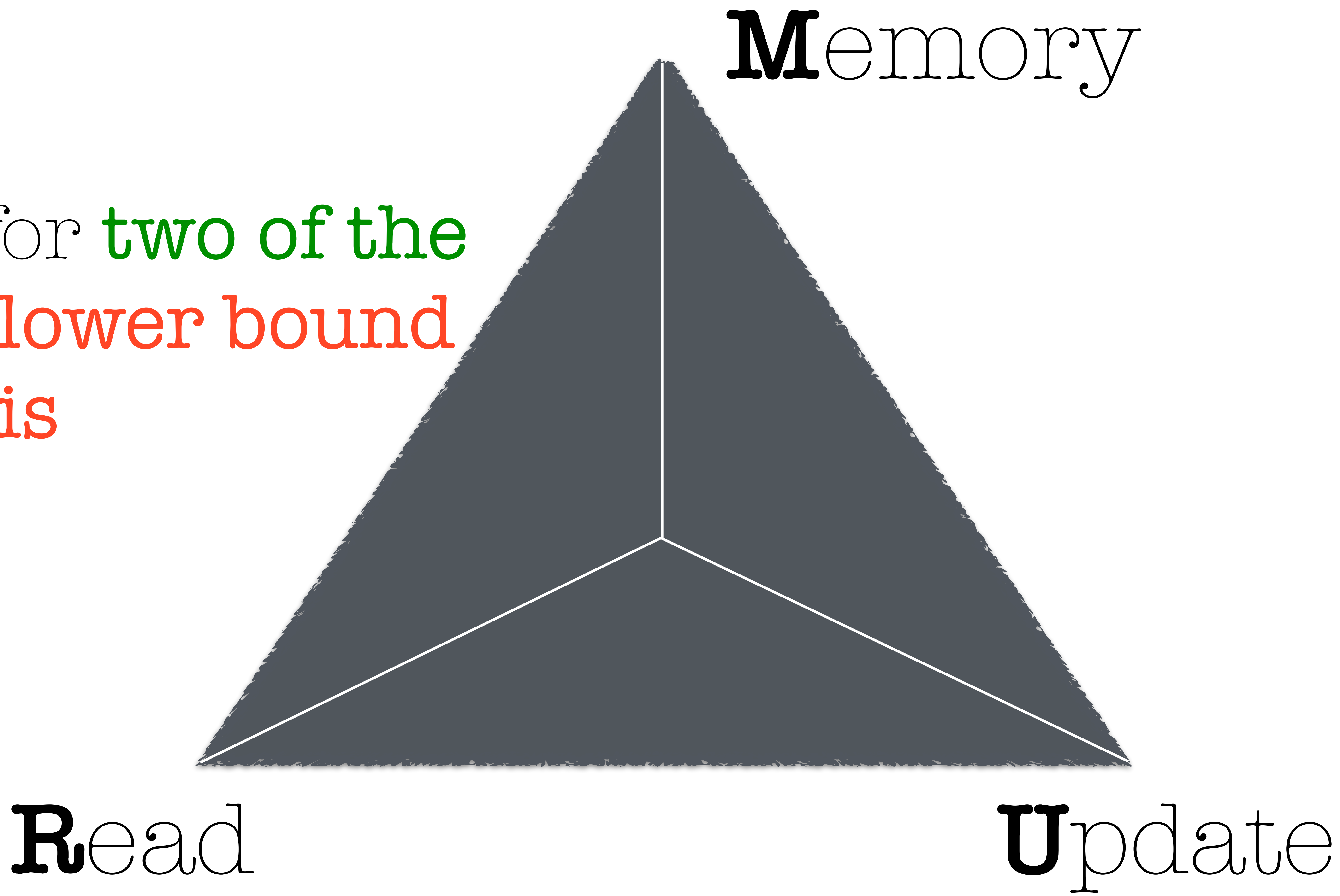


RUM conjecture

A three-way tradeoff

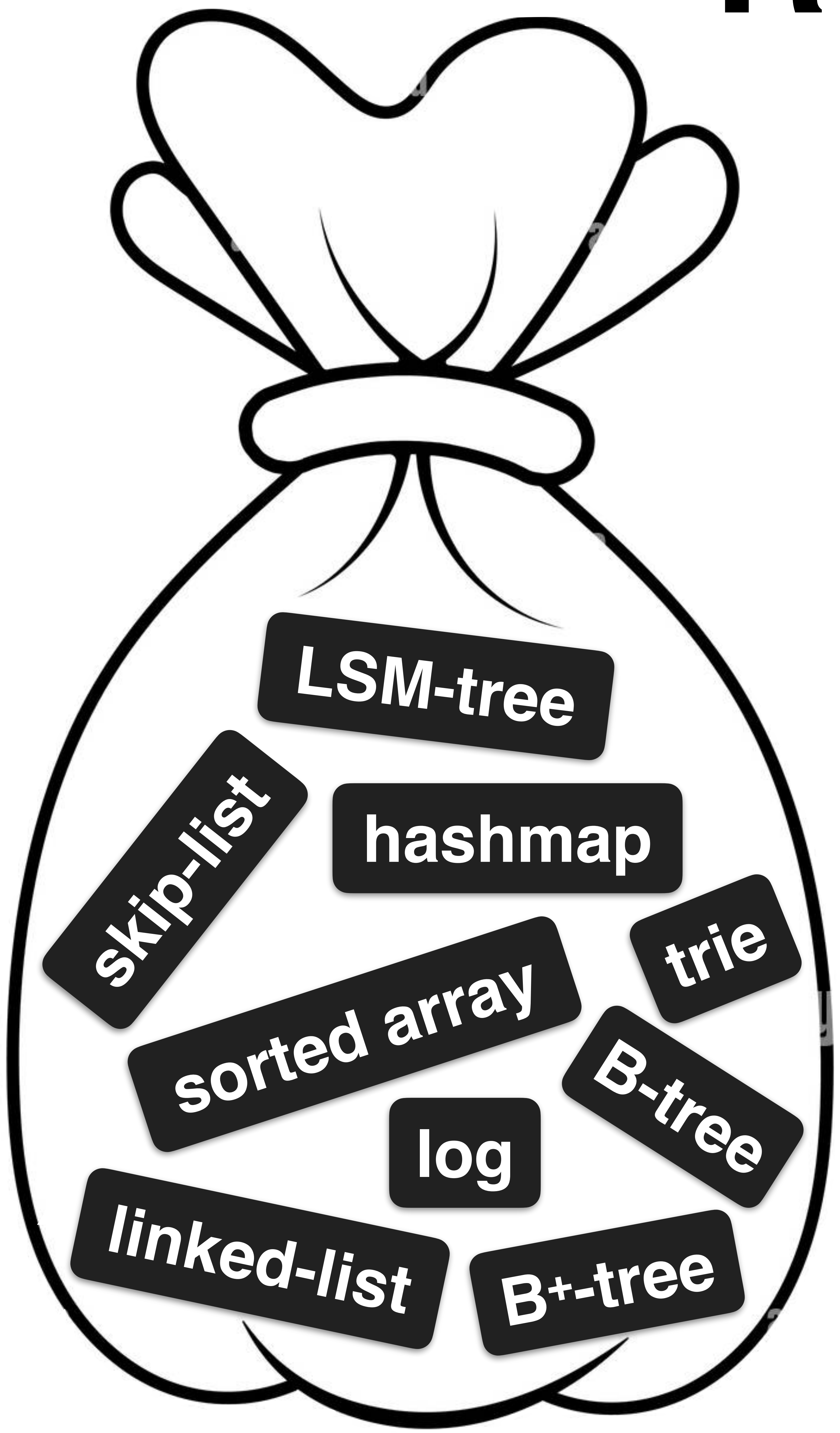
“

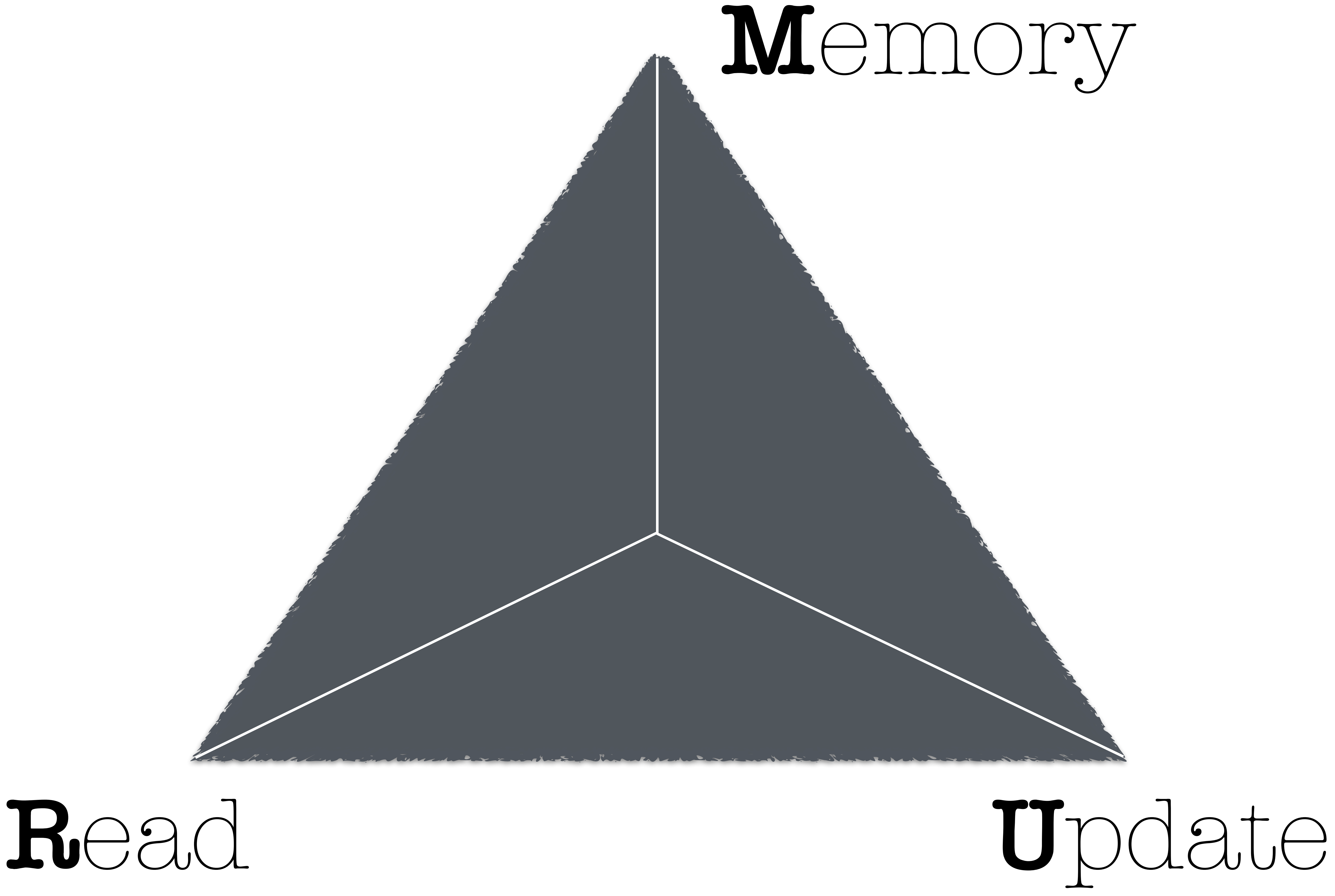
... setting an upper bound for two of the RUM axes, implies a hard lower bound for the third axis



RUM conjecture

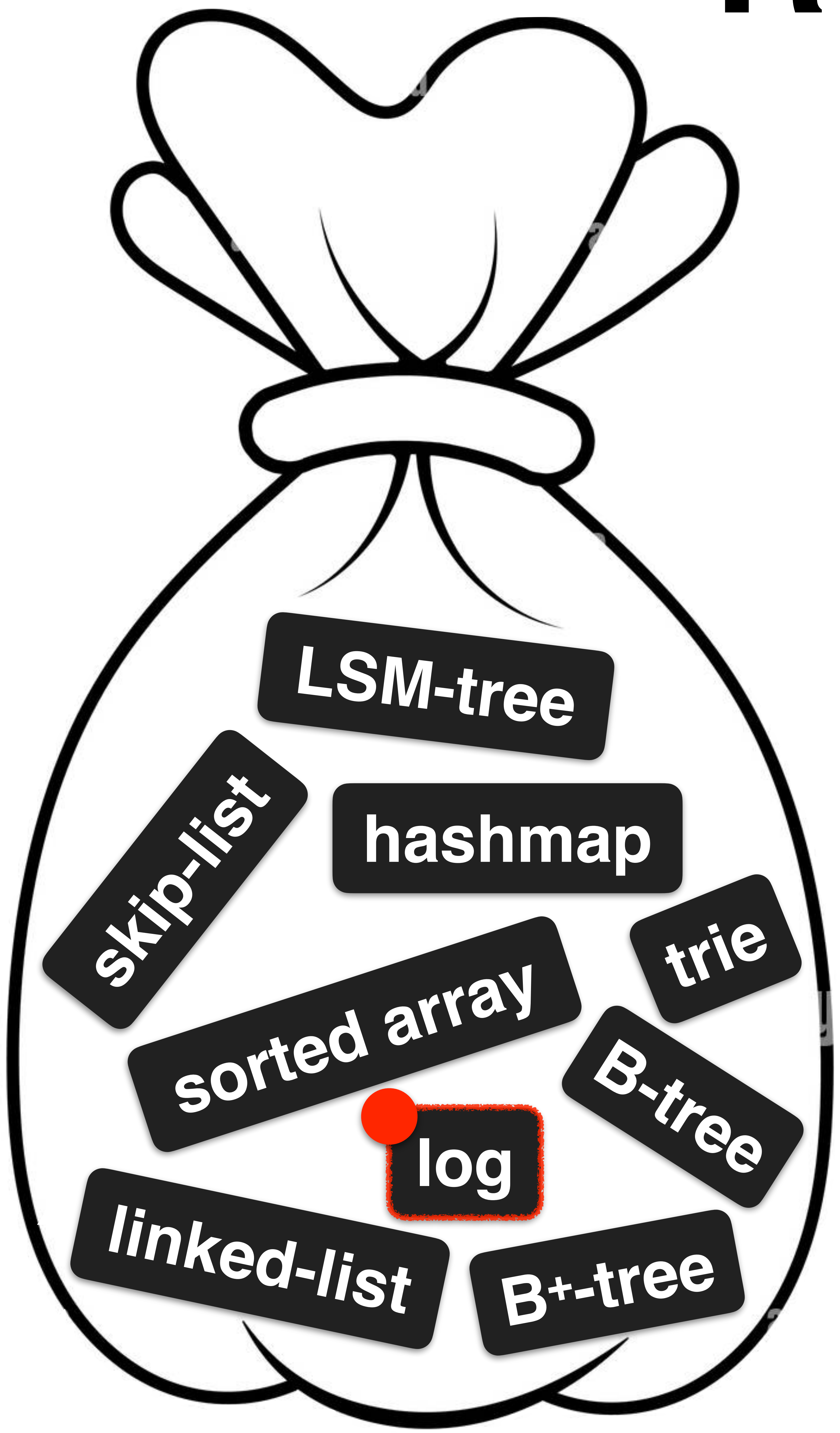
A three-way tradeoff

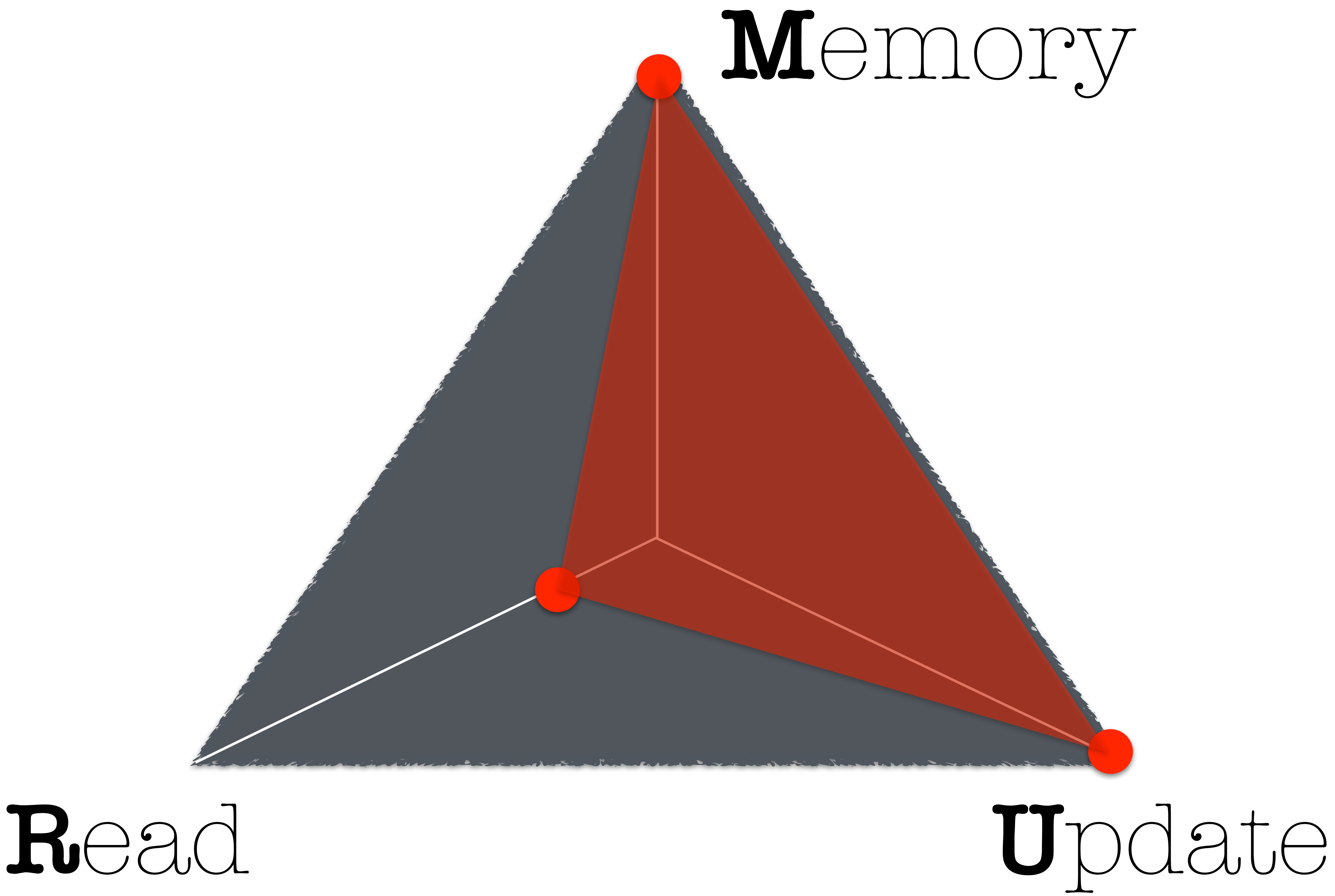




RUM conjecture

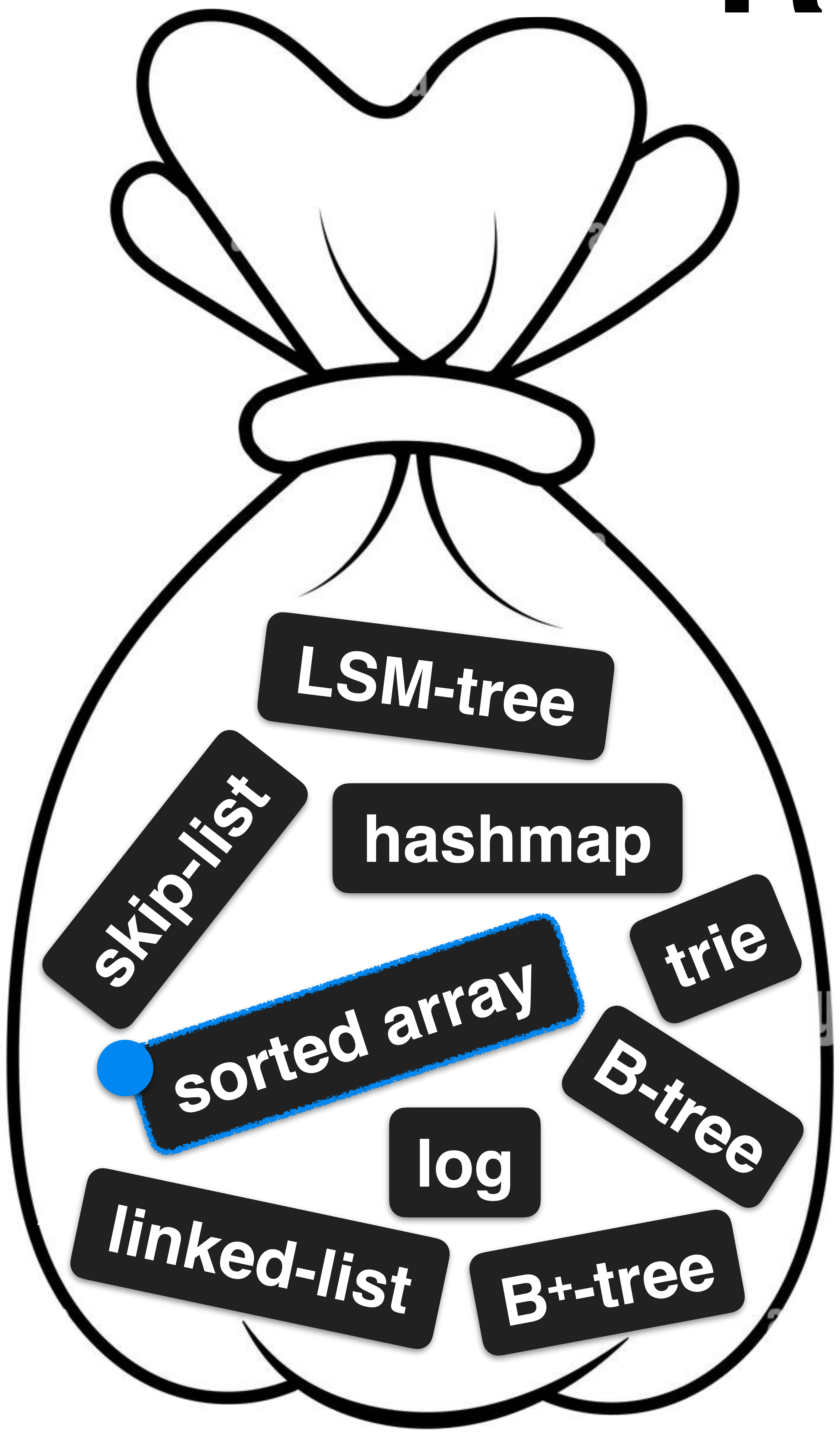
A three-way tradeoff

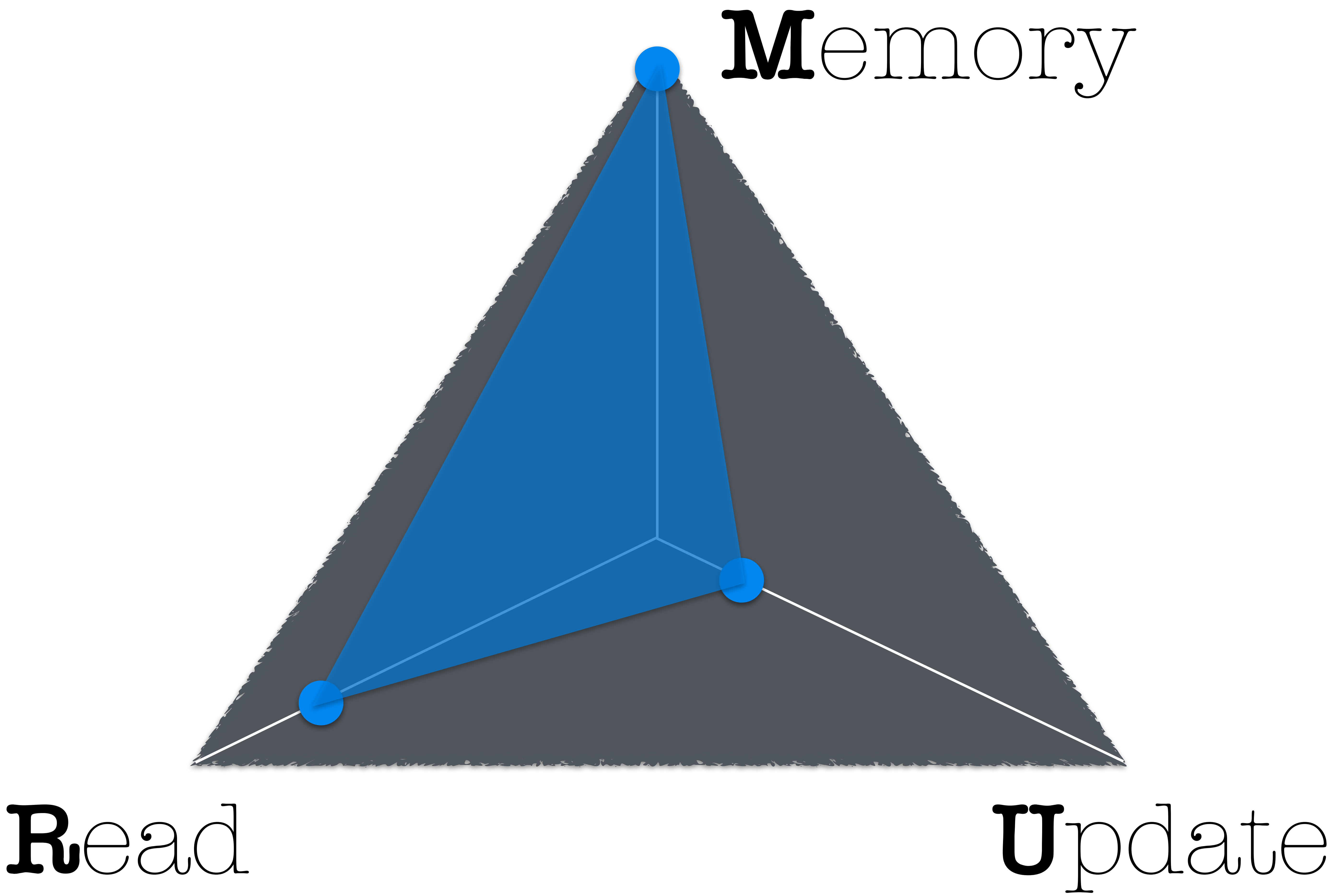




RUM conjecture

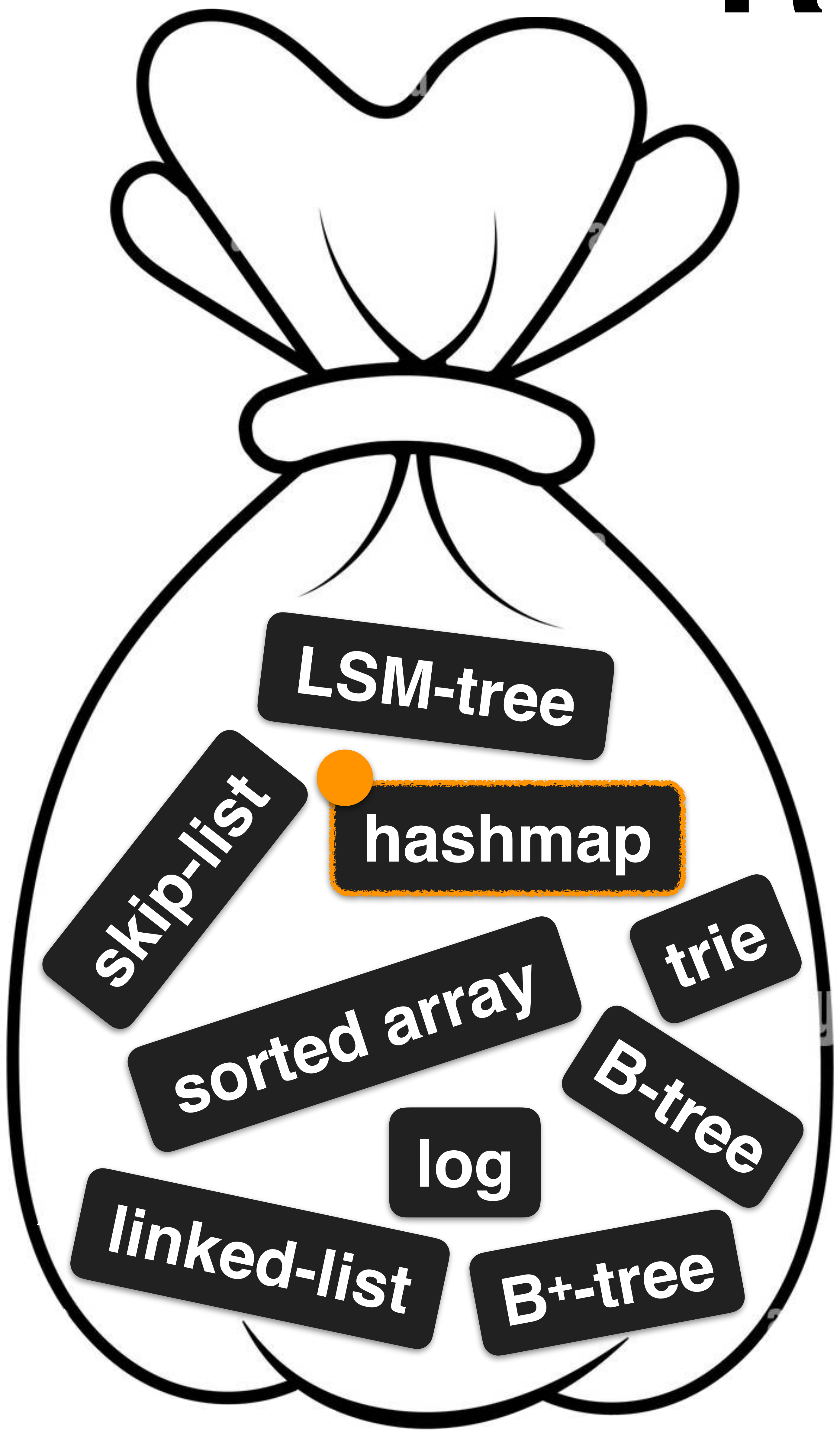
A three-way tradeoff

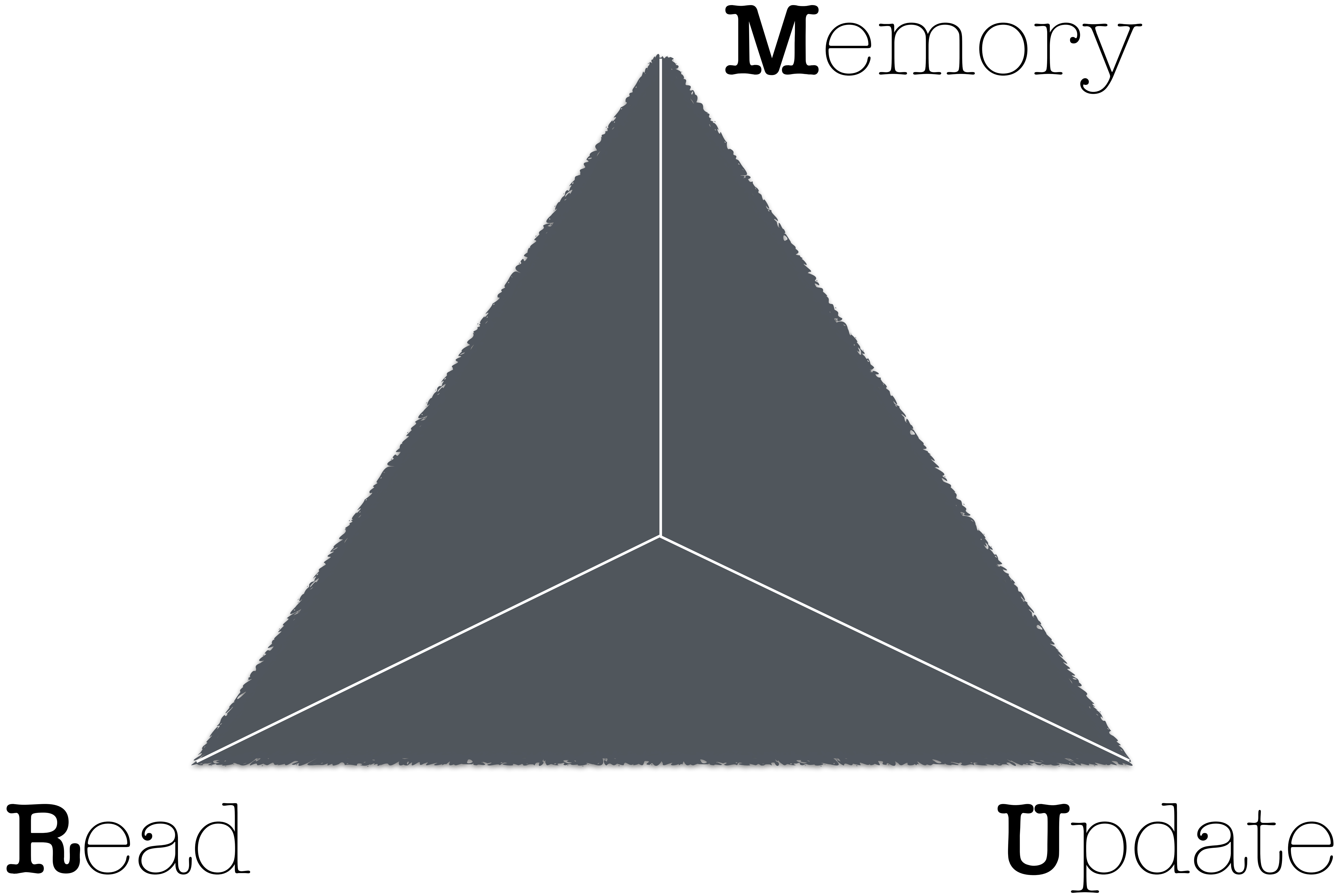




RUM conjecture

A three-way tradeoff





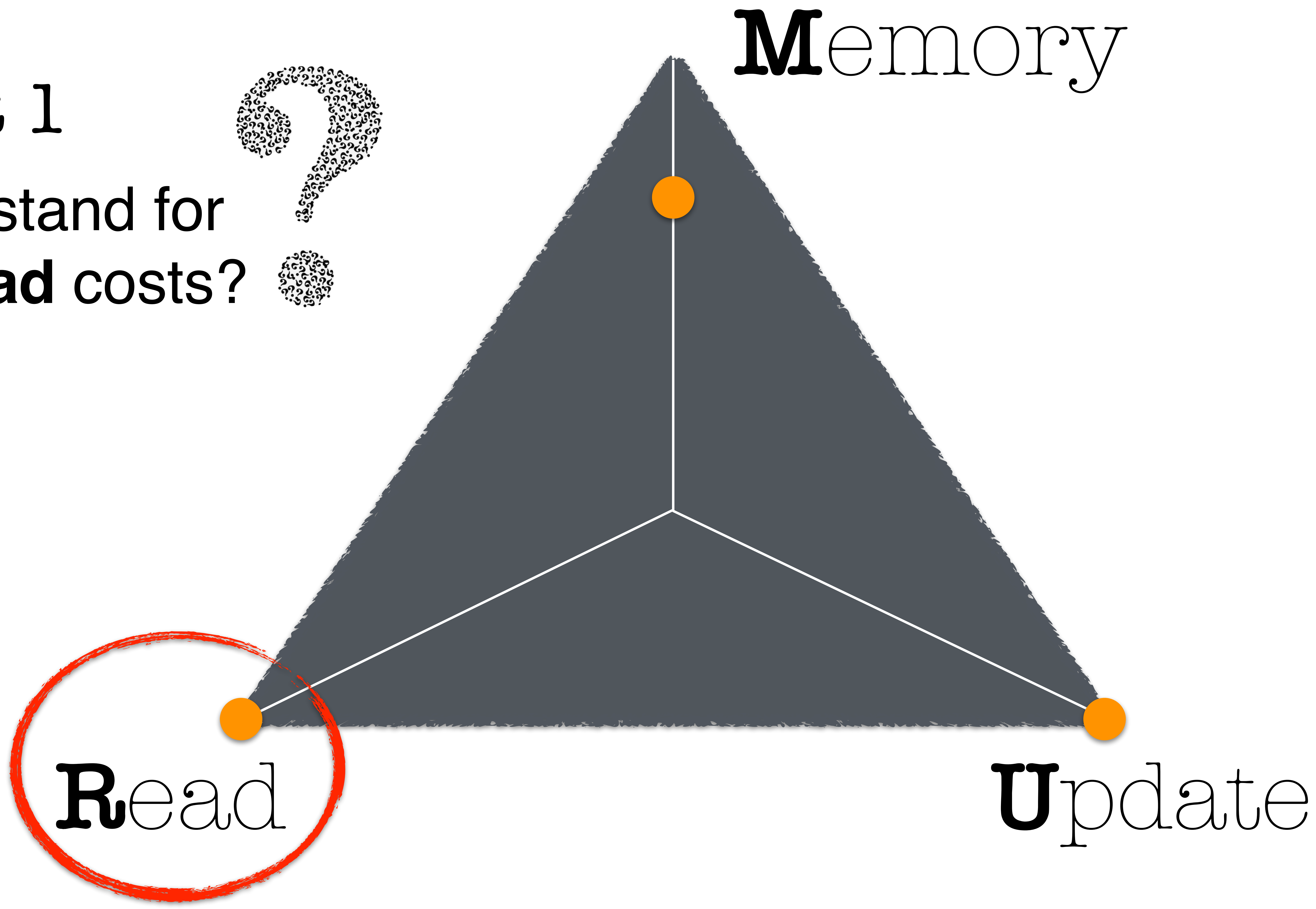
RUM conjecture

A three-way tradeoff

Thought Experiment 1

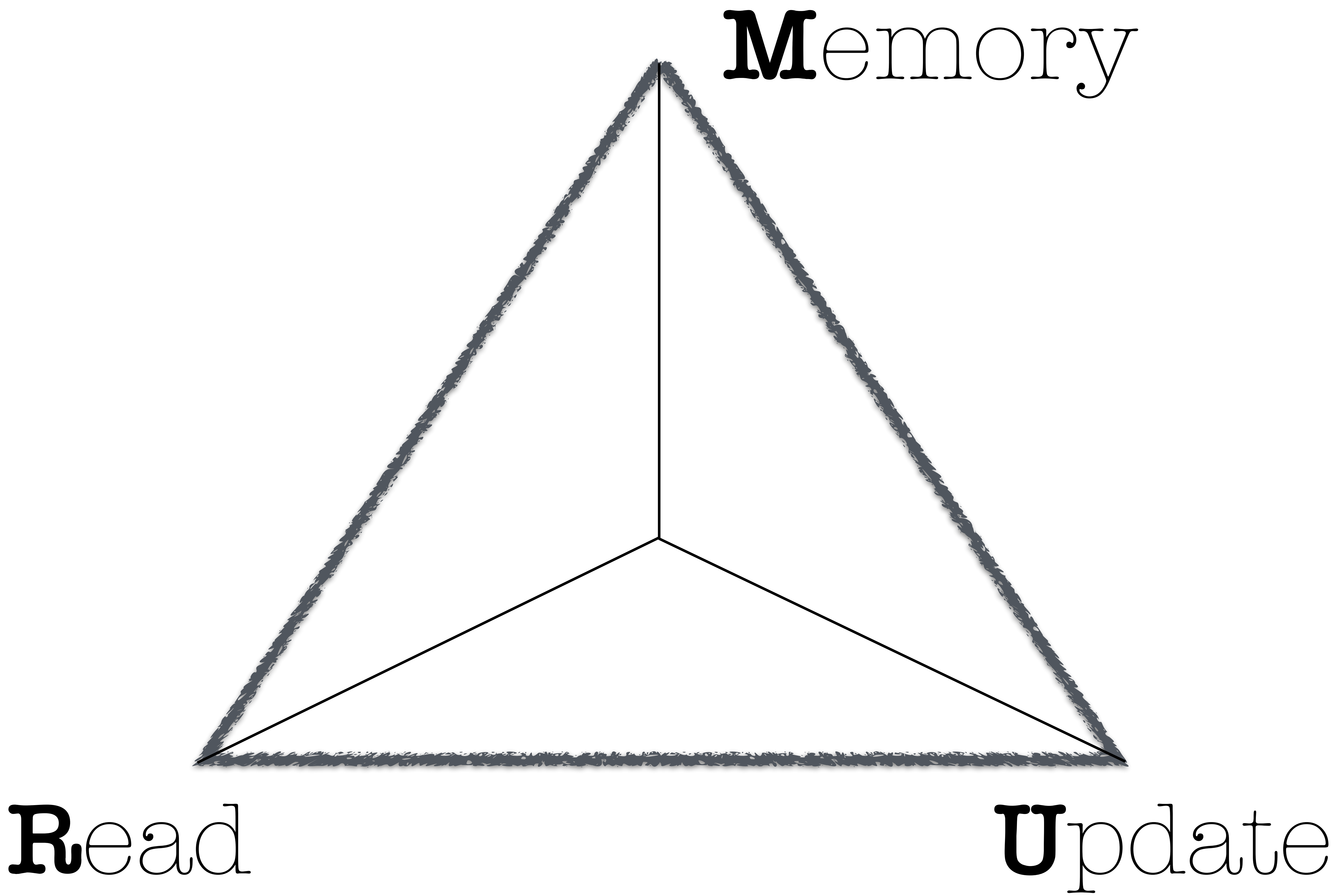
Where does a **hashmap** stand for **memory**, **update**, and **read** costs?

point read
vs
range read



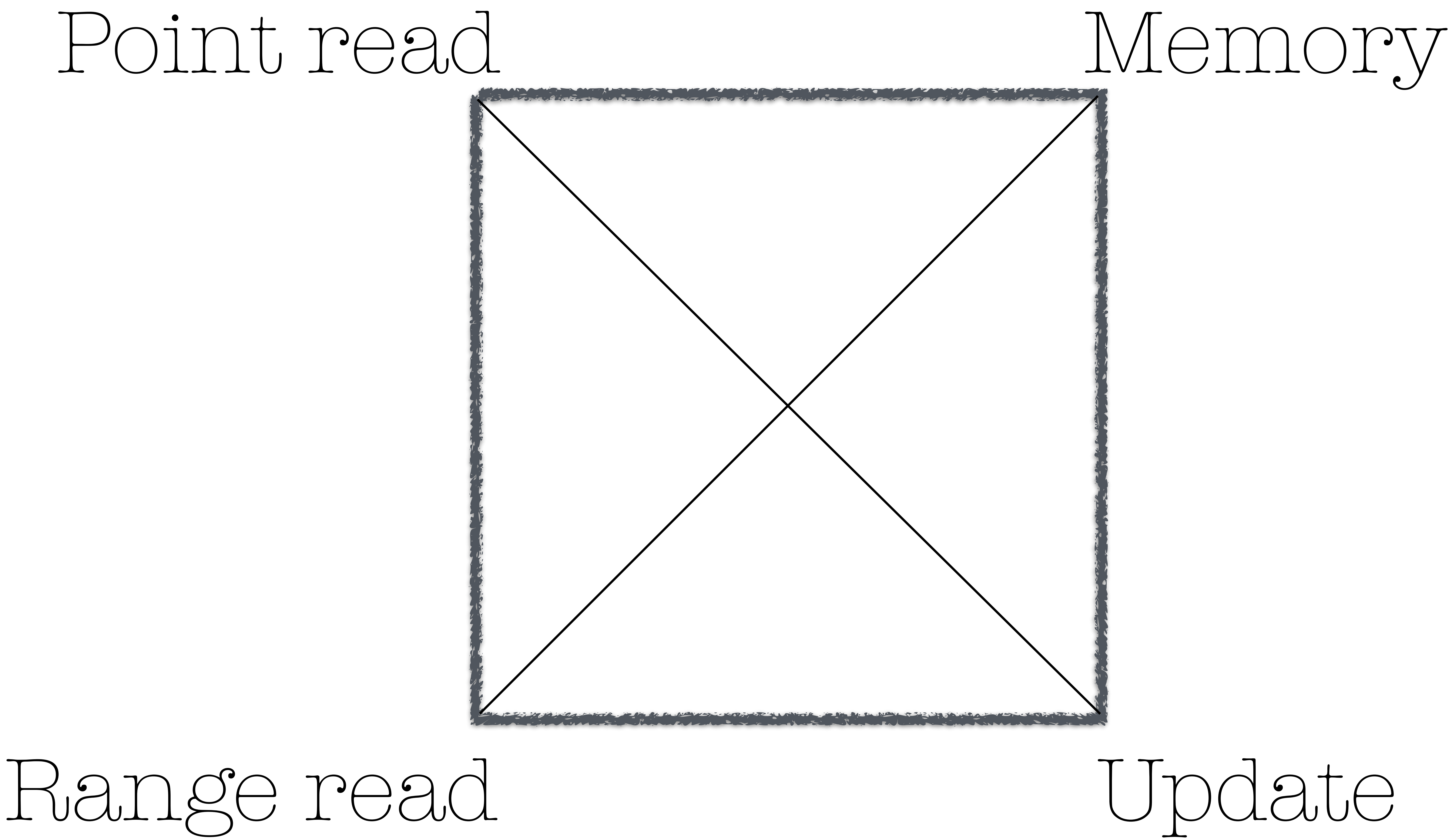
RUM conjecture

A three-way tradeoff



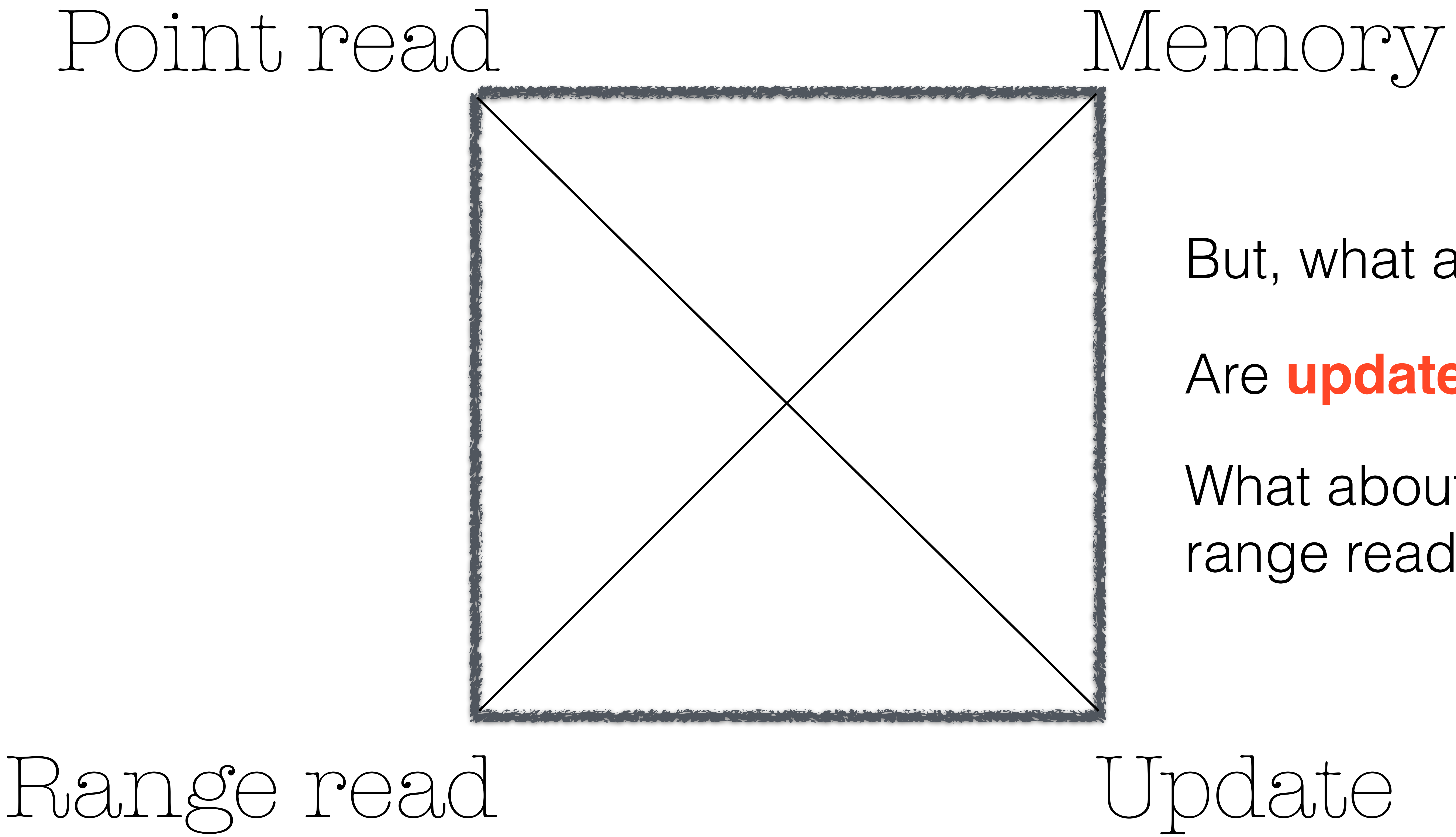
Extending the **RUM tradeoff**

A multi-way tradeoff



Extending the **RUM tradeoff**

A multi-way tradeoff



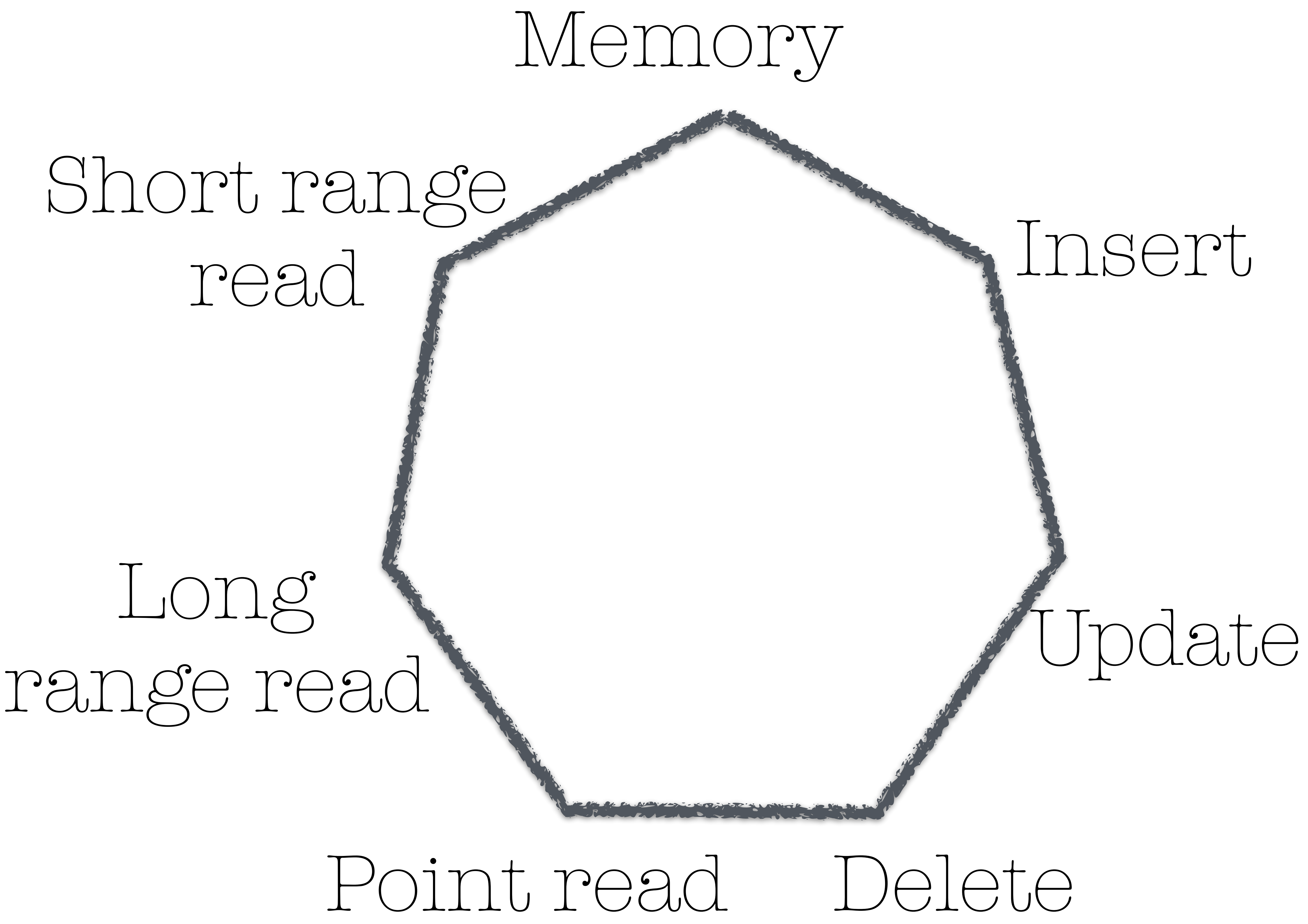
But, what about **deletes**?

Are **updates** the same as **inserts**?

What about **selectivity** of range reads?

Extending the **RUM tradeoff**

A multi-way tradeoff



But, what about **deletes**?

Are **updates** the same as **inserts**?

What about **selectivity** of range reads?

cloud
cost

hardware

performance
tradeoffs

Designing data systems = HARD PROBLEM

index
design

access method

application
requirements

Designing data systems

Solving a hard problem

Ask for the HiPPO

Highest Paid Person's Opinion

Designing data systems

Solving a hard problem

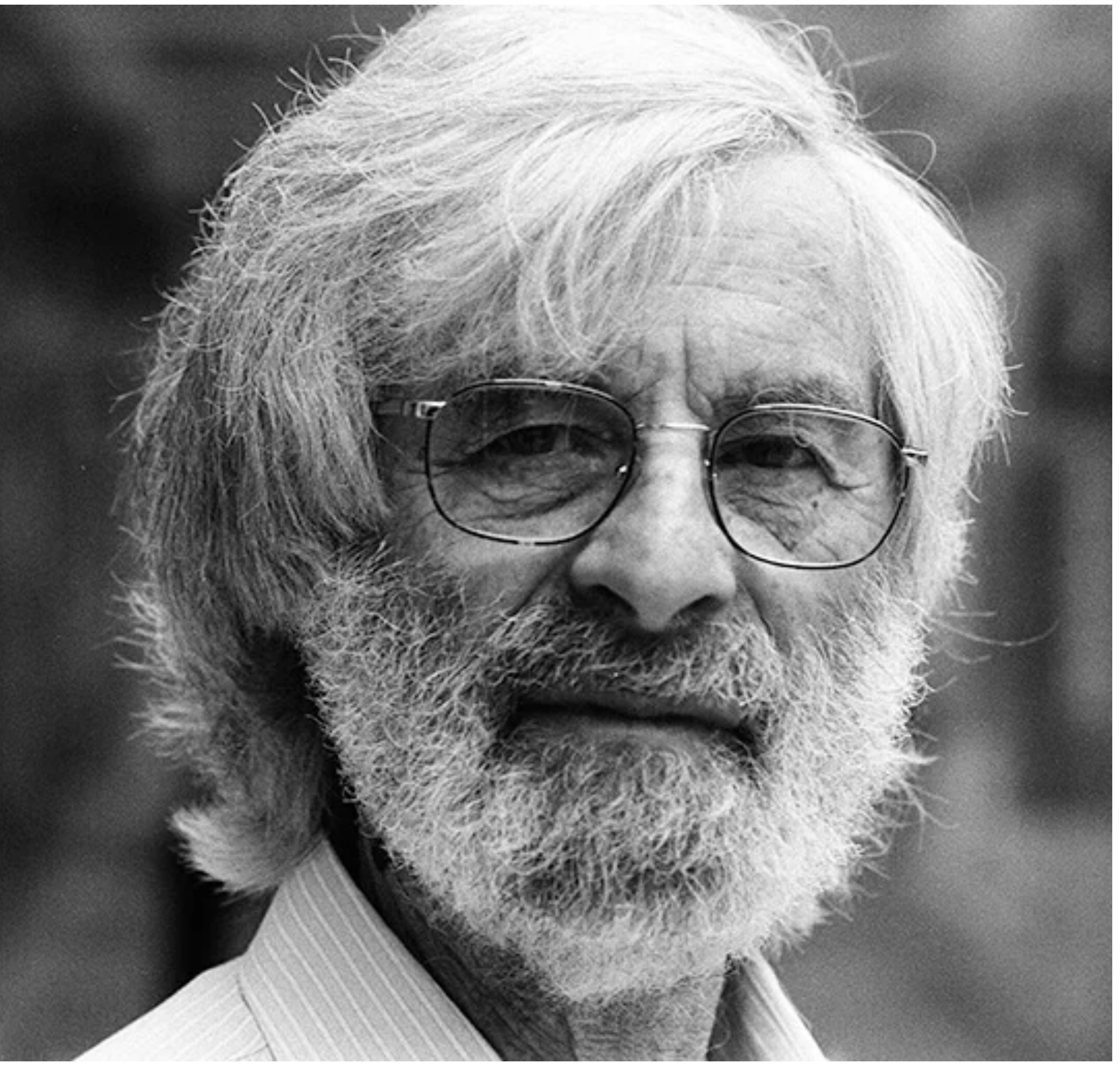
No one knows everything!

Ask for the HiPPO

Highest Paid Person's Opinion

Designing data systems

Solving a hard problem



Lesley Lamport, Microsoft Research

MA ’63, PhD ’72 Brandeis University

ACM Turing Award 2013

No one knows everything!

The Part-Time Parliament

LESLIE

Digital E

Recent ar
despite t
consistent
and the f
way of im

Categorie
uted Syst

Paxos Made Simple

Leslie Lamport

01 Nov 2001

Abstract

The Paxos algorithm, when presented in plain English, is very simple.

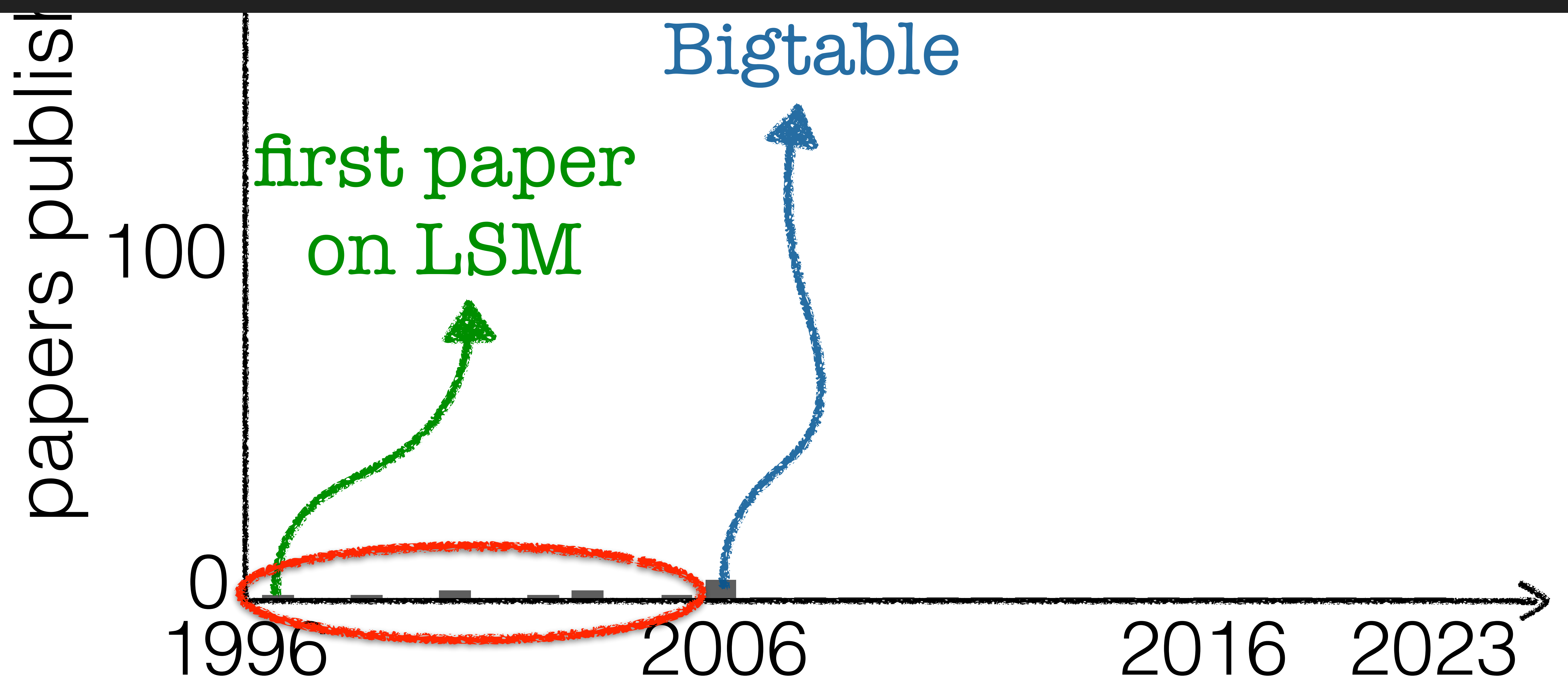
Brandeis UNIVERSITY

Designing data systems

Solving a hard problem

Patrick O'Neil, UMass Boston
Co-inventor of the LSM-tree

No one knows everything!

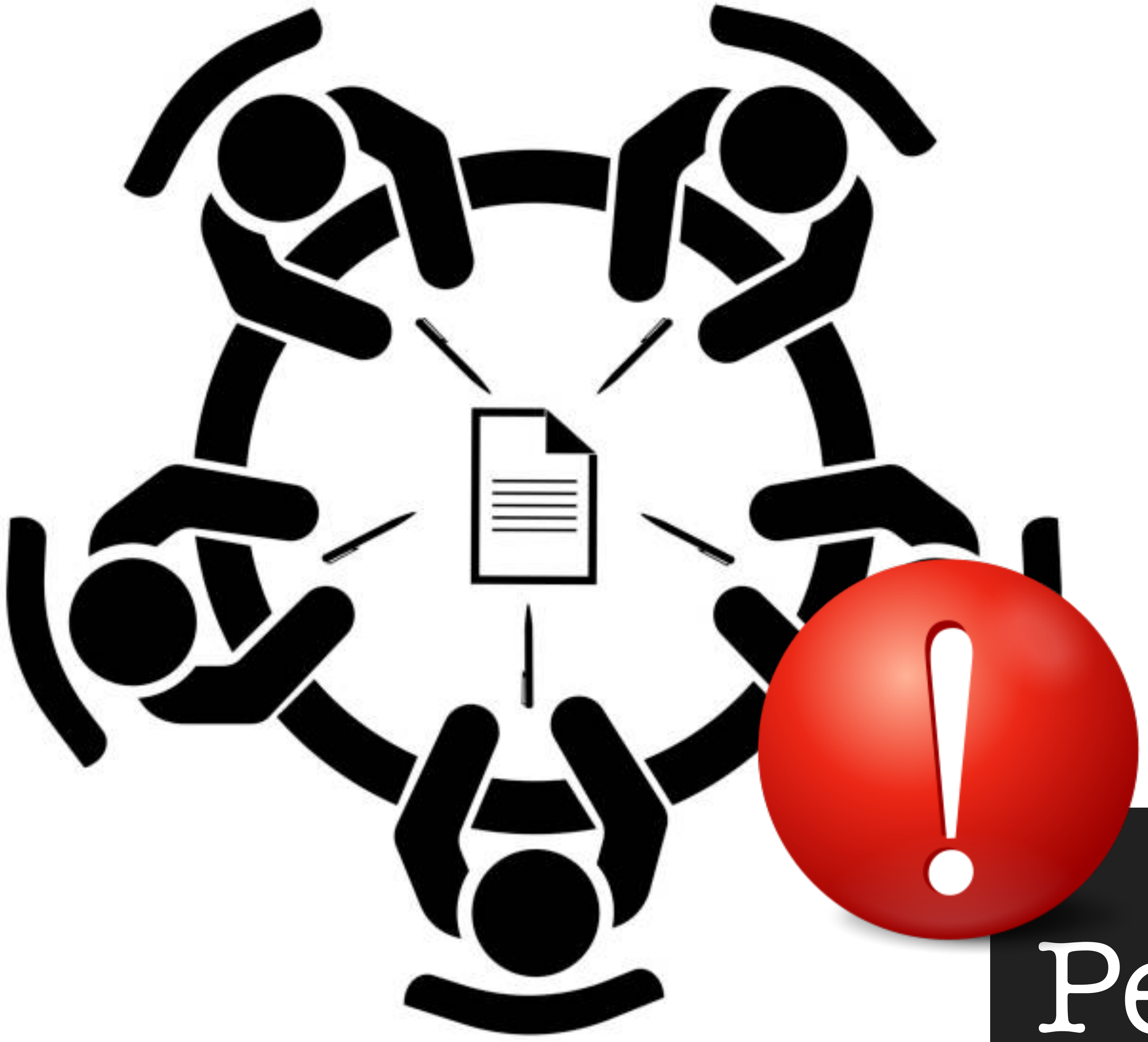


* data from Google scholar

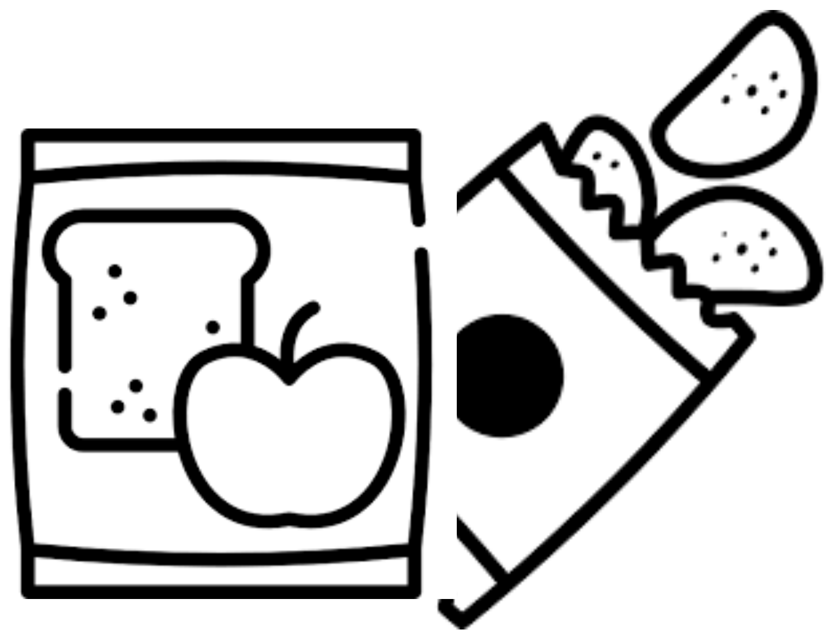
Brandeis
UNIVERSITY

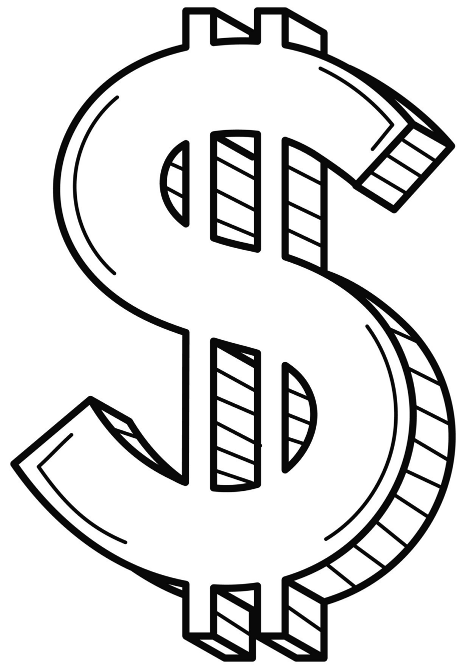
Designing data systems

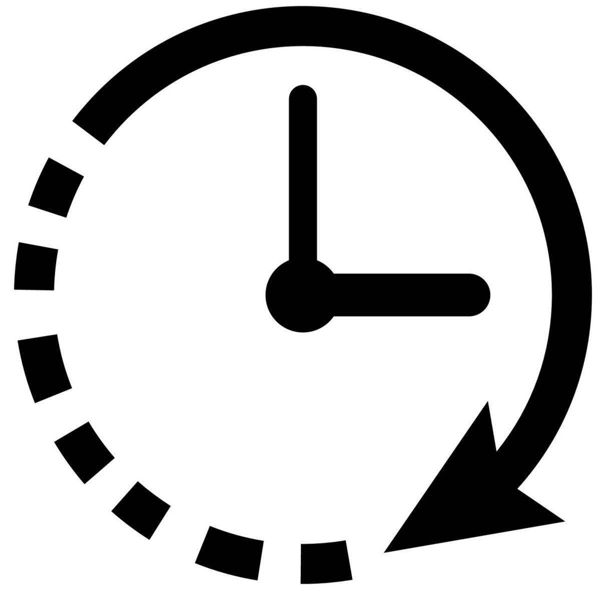
Solving a hard problem



Get the COSI 167 students







Perfectly tuned data system!

Designing data systems

Solving a hard problem

Manual designing &
hand-tuning hundreds of knobs
do NOT SCALE

Designing data systems

Solving a hard problem

Manual designing &
hand-tuning hundreds of knobs
do NOT SCALE

Self-designing, self-tuning, and
adaptive data systems

Designing data systems

Solving a hard problem

Self-designing, self-tuning, and adaptive data systems

Adaptive data layouts: row stores vs. column stores vs. hybrids

Self-tuning LSM-engines

Adaptive (**adaptive**) indexing

Workload-aware data re-organization

Self-designing storage engines

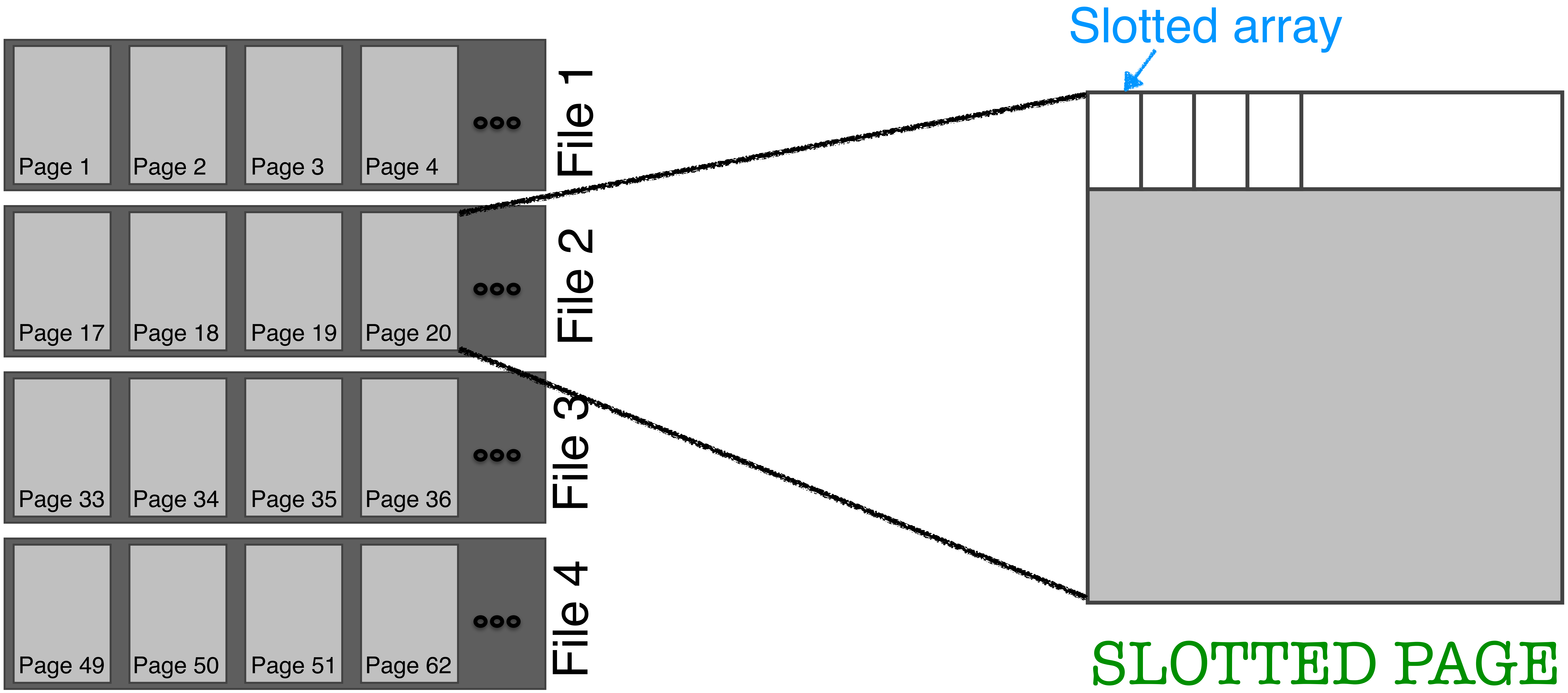
Hardware-conscious memory manager

Cloud cost-optimized data systems

Understanding **data placement**

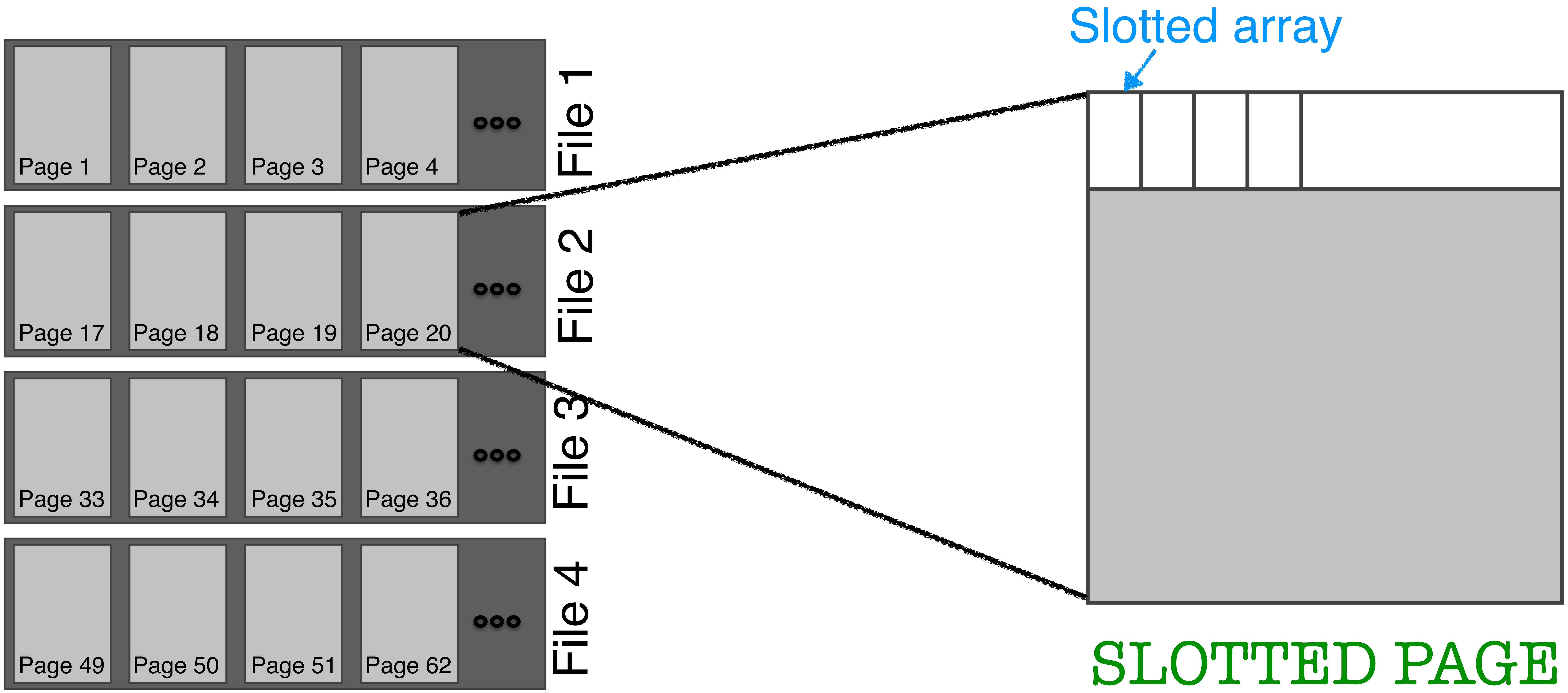
Understanding **data placement**

Files, pages, tuples



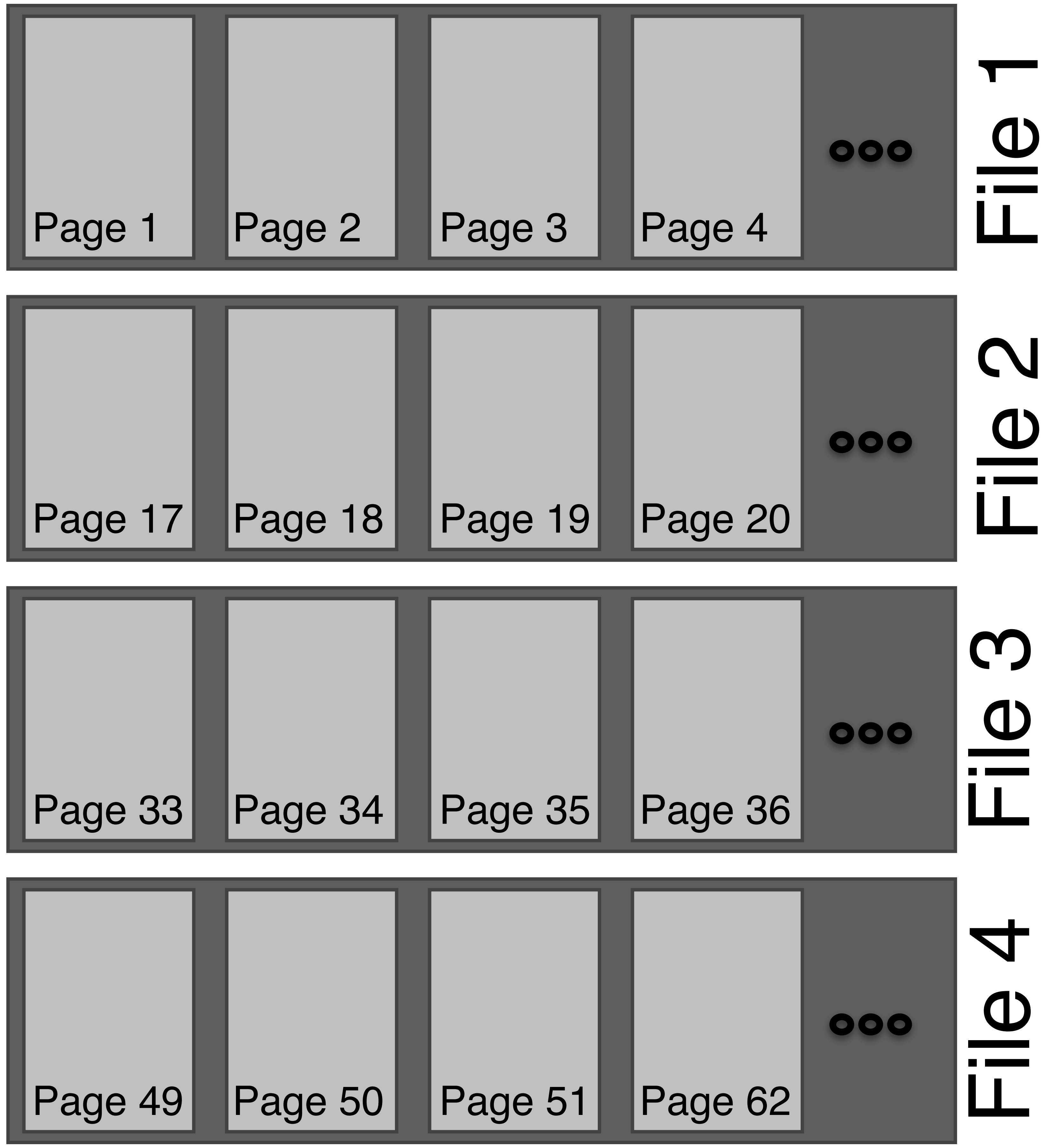
Understanding **data placement**

Files, pages, tuples

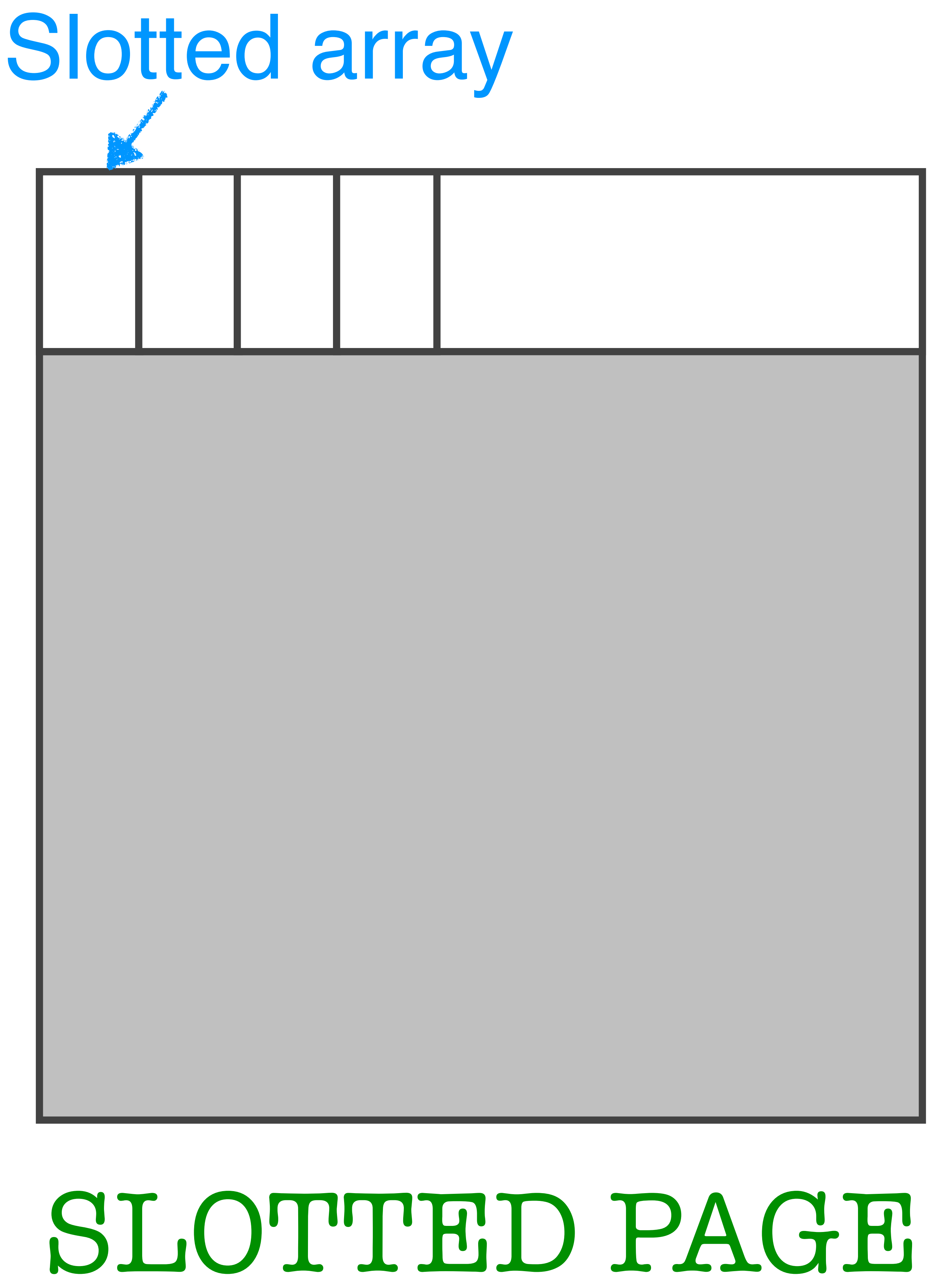


Understanding **data placement**

Files, pages, tuples

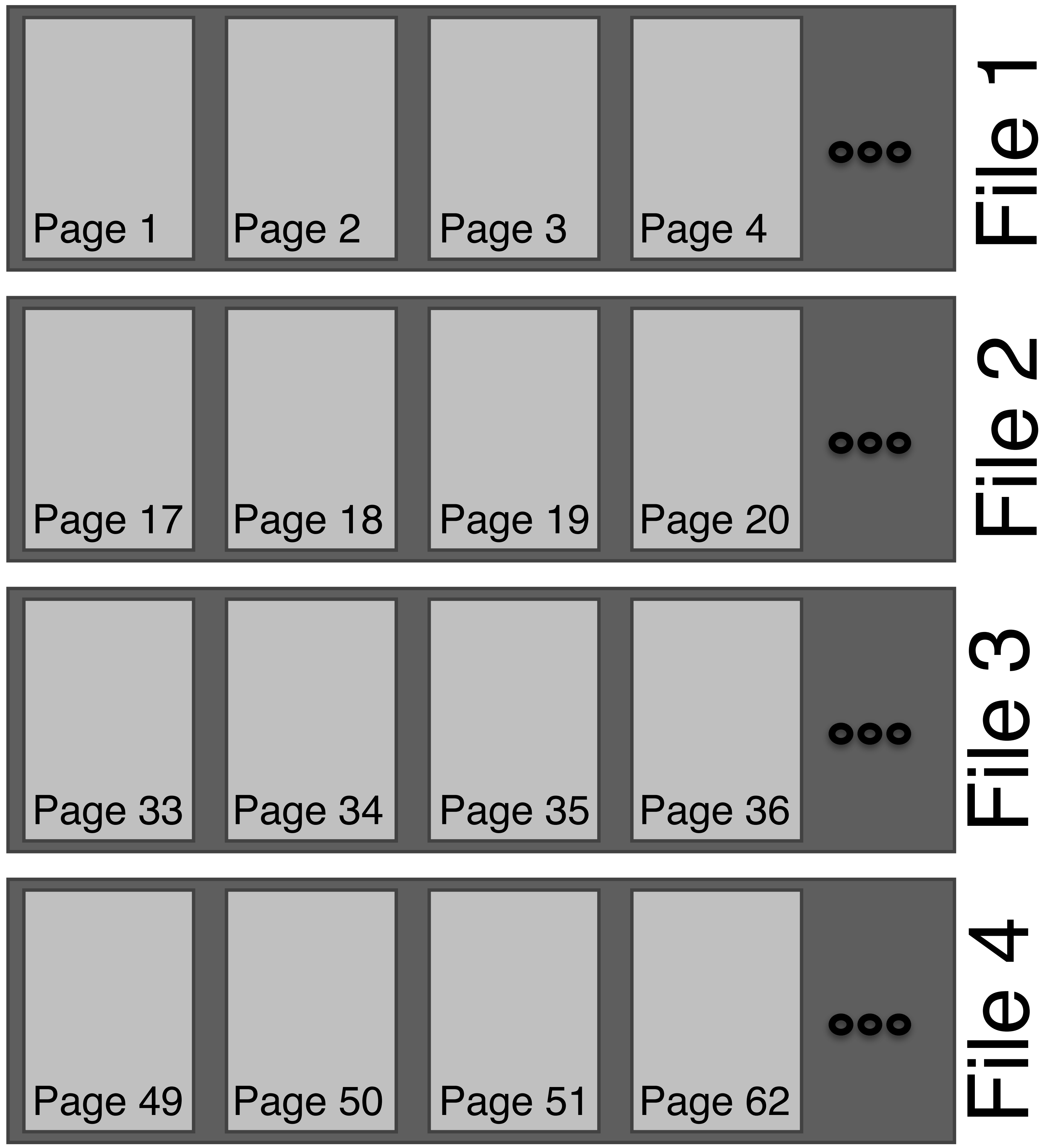


insert(Tuple 1)

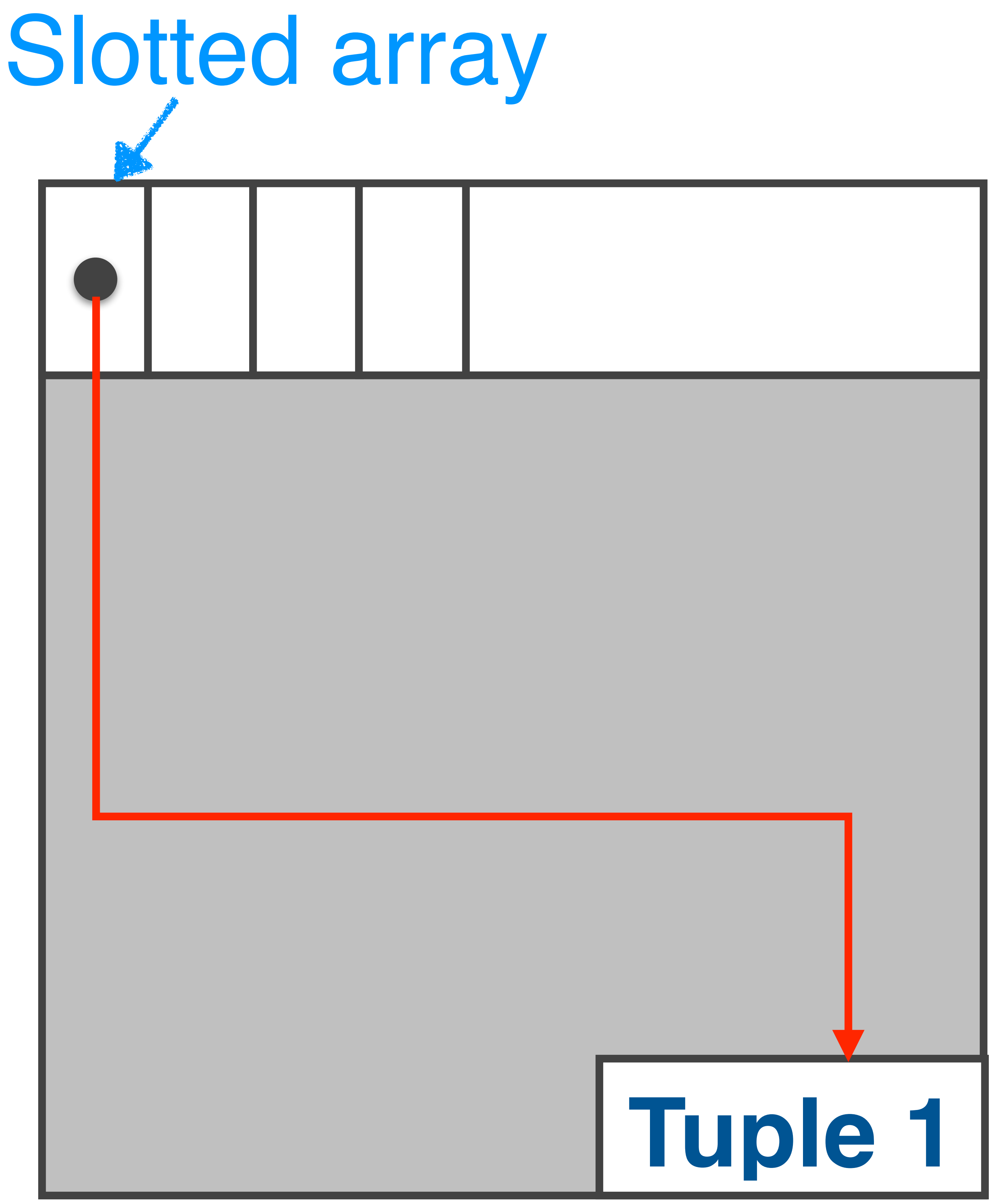


Understanding **data placement**

Files, pages, tuples



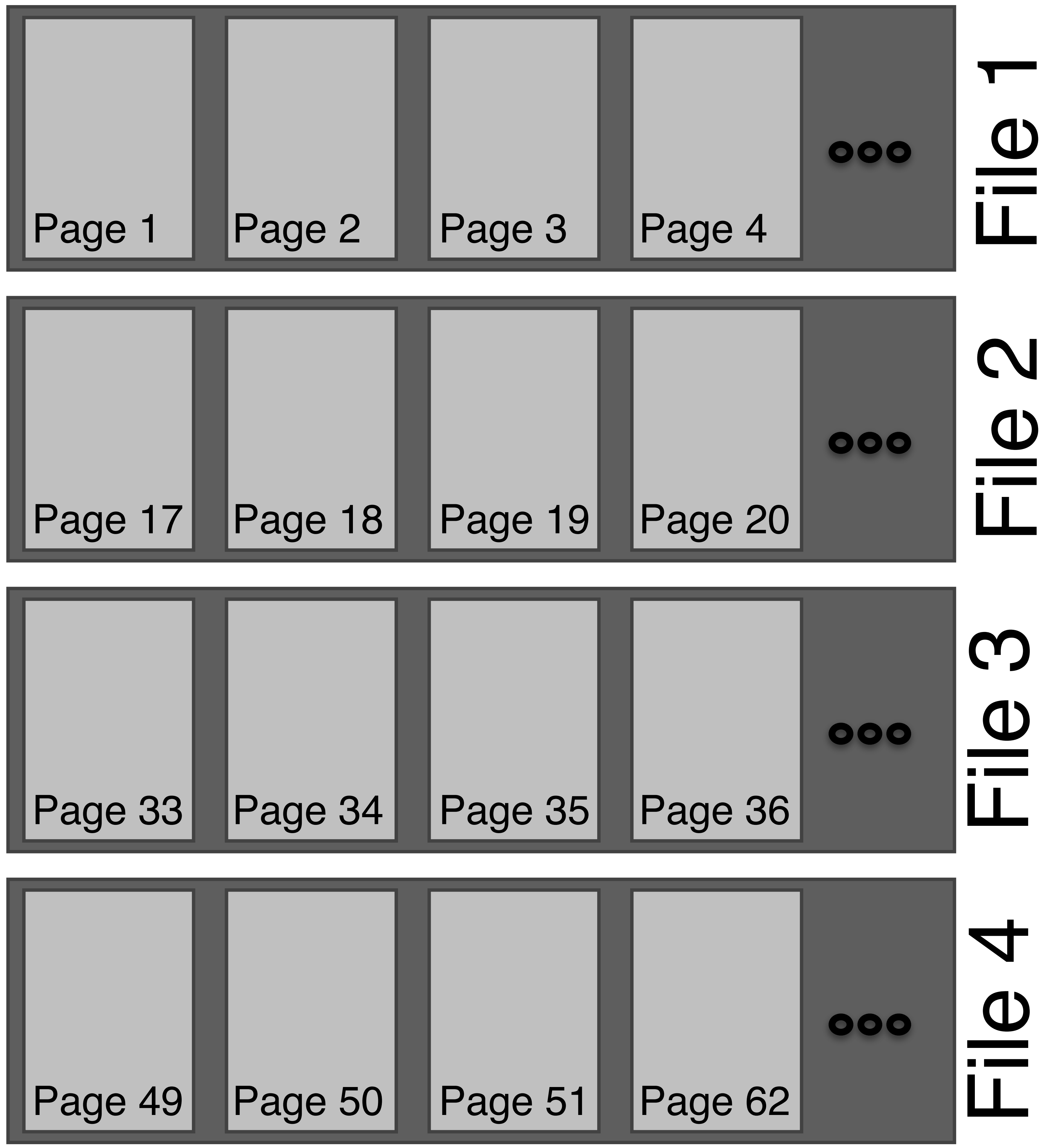
insert(Tuple 1)



SLOTTED PAGE

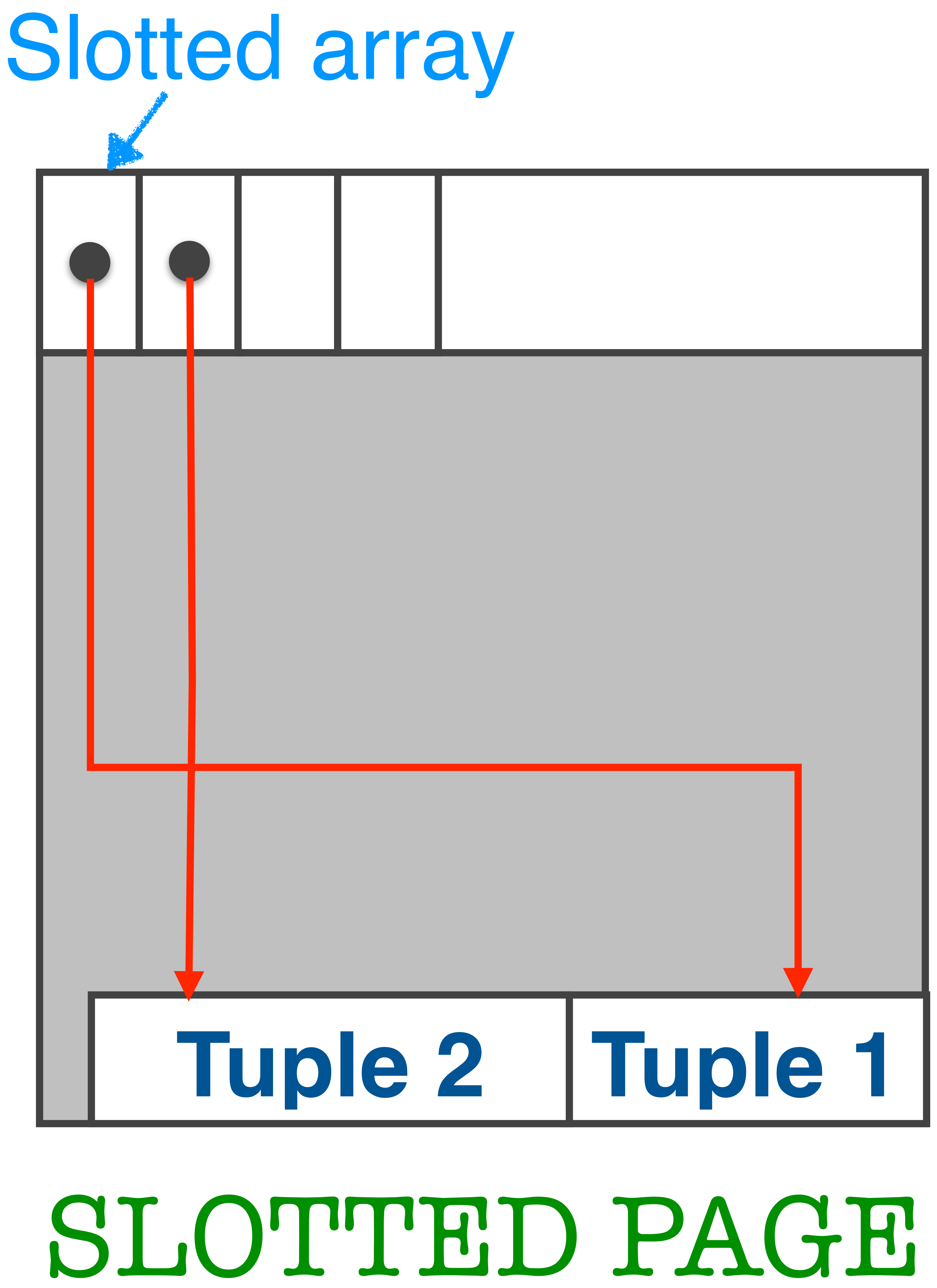
Understanding **data placement**

Files, pages, tuples



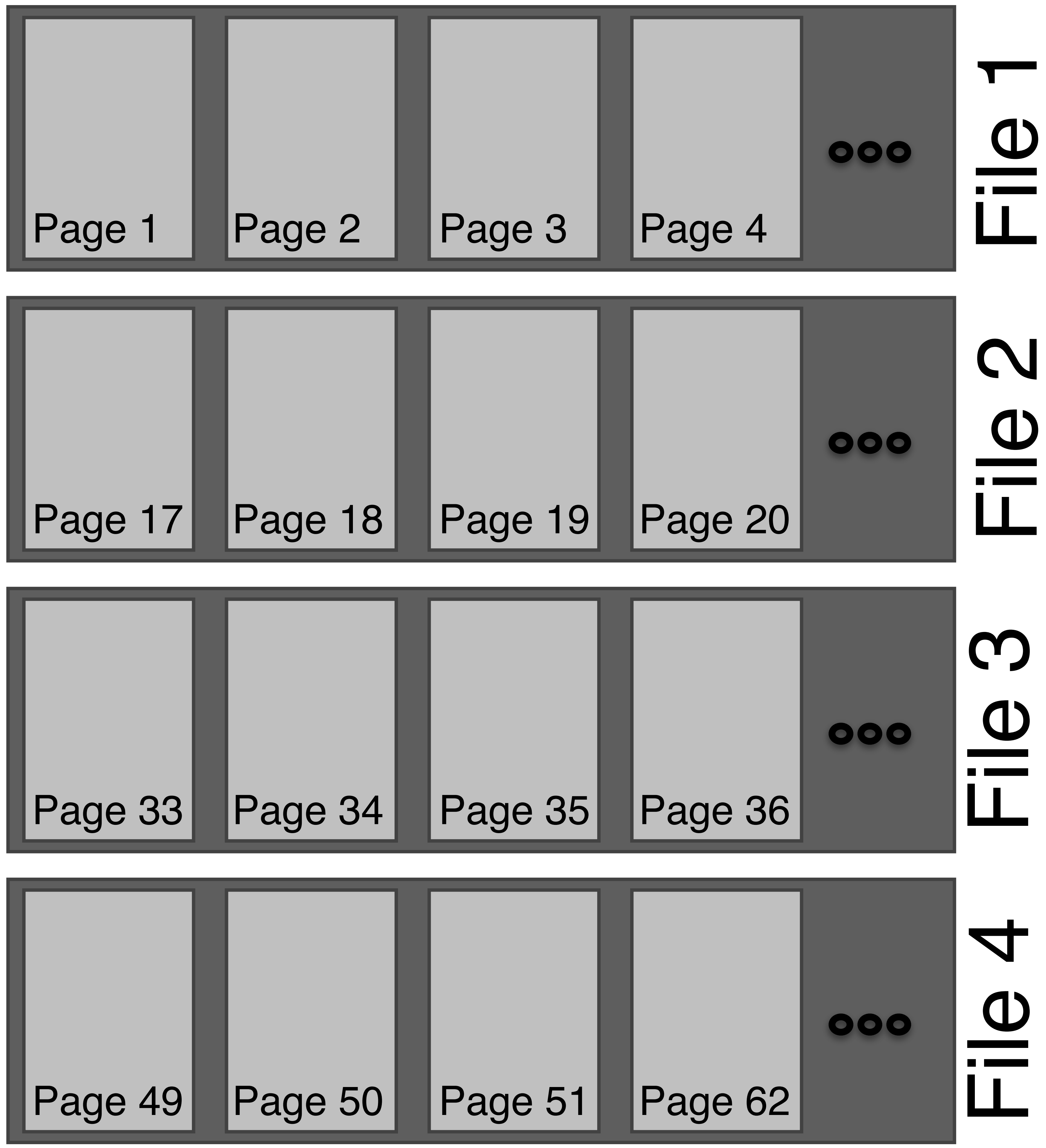
insert(Tuple 1)

insert(Tuple 2)

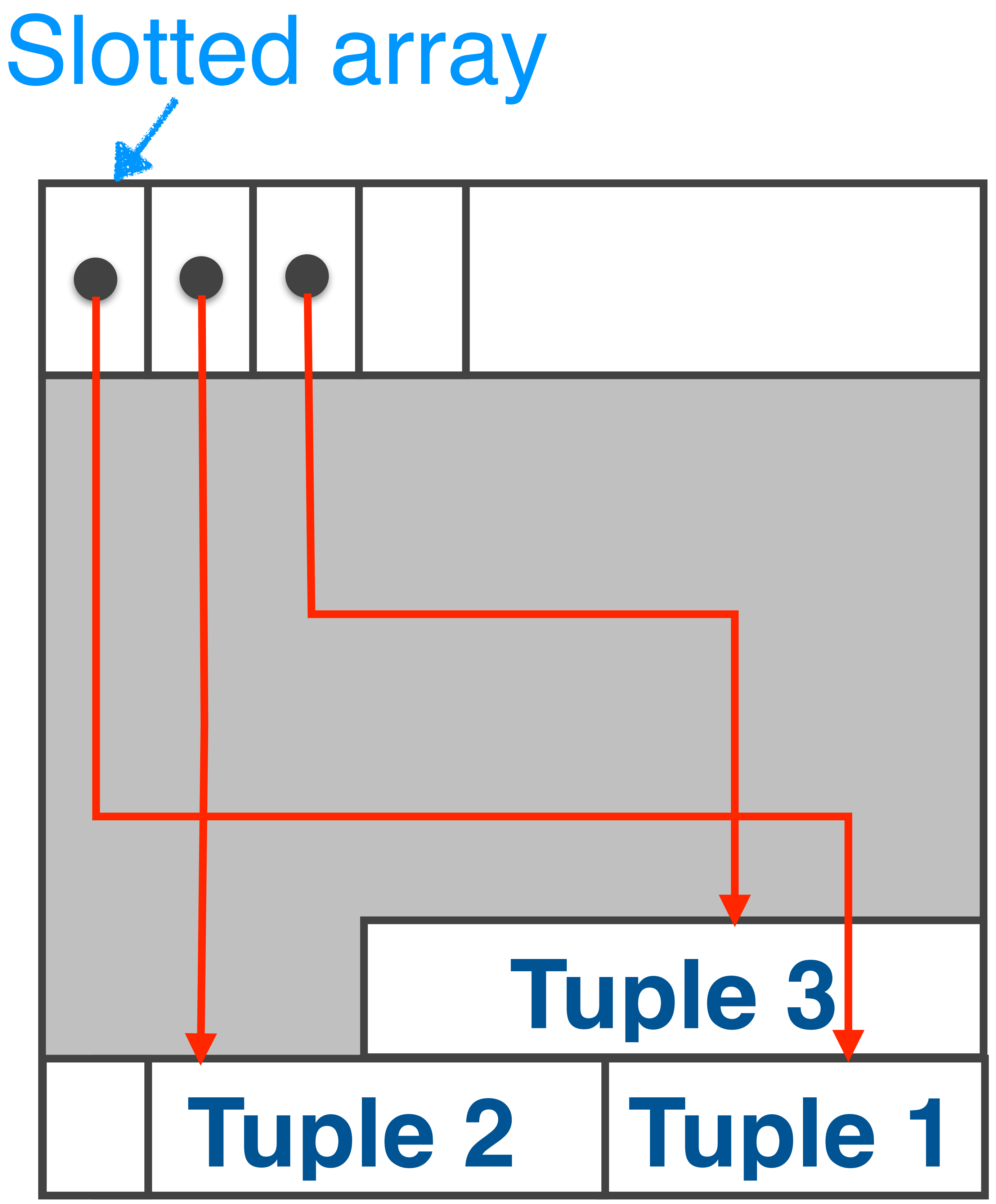


Understanding **data placement**

Files, pages, tuples



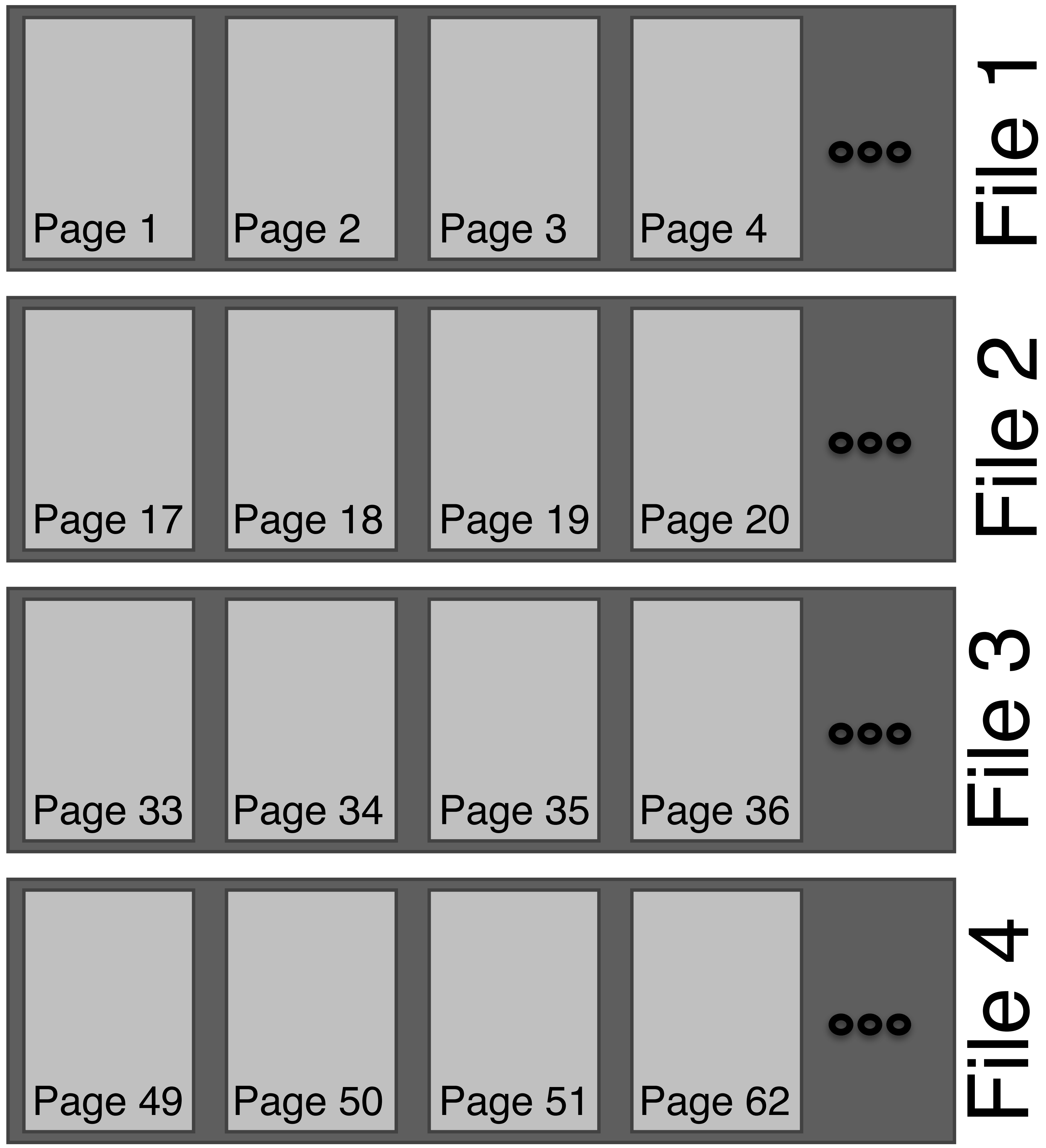
insert(Tuple 1)
insert(Tuple 2)
insert(Tuple 3)



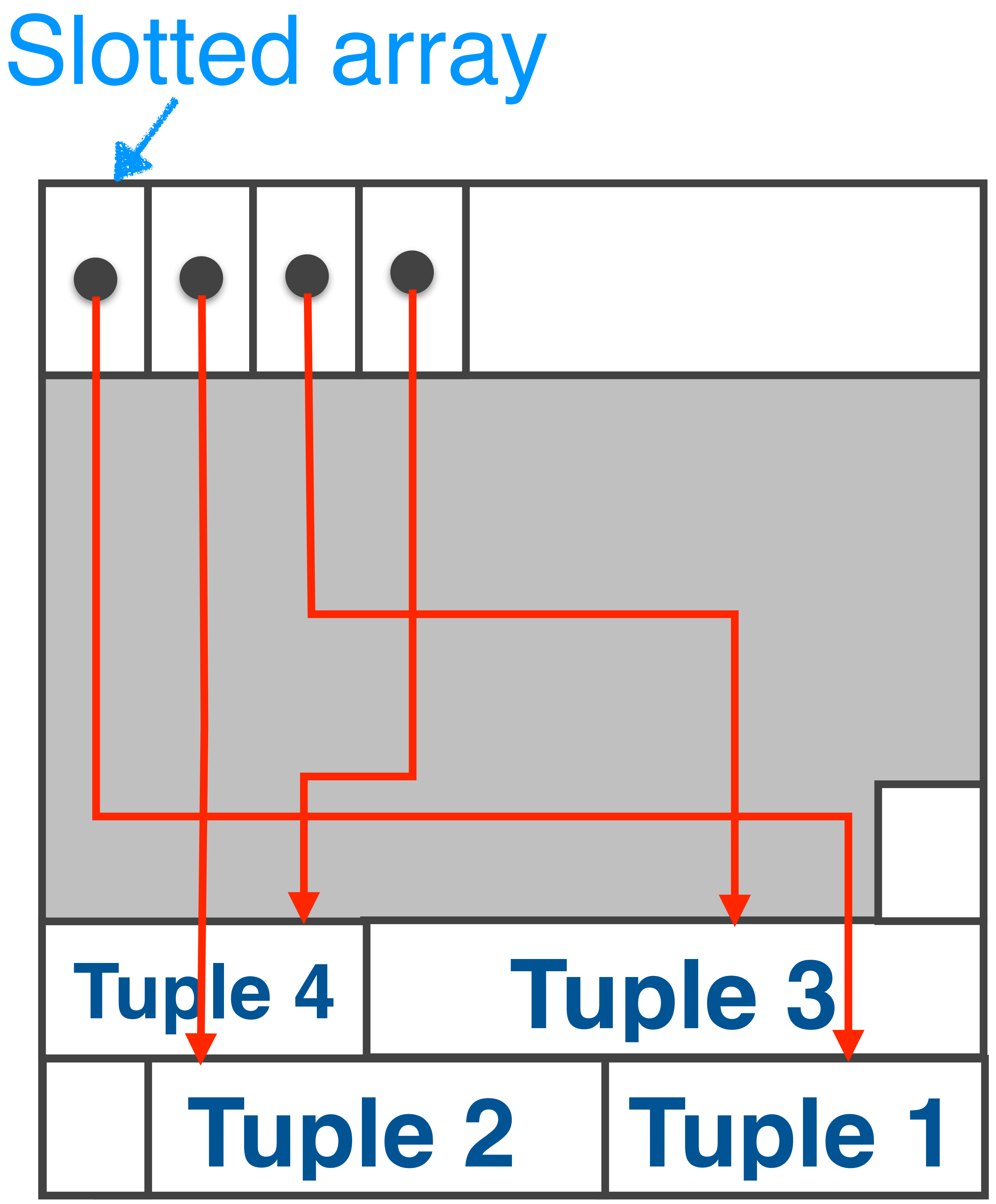
SLOTTED PAGE

Understanding **data placement**

Files, pages, tuples



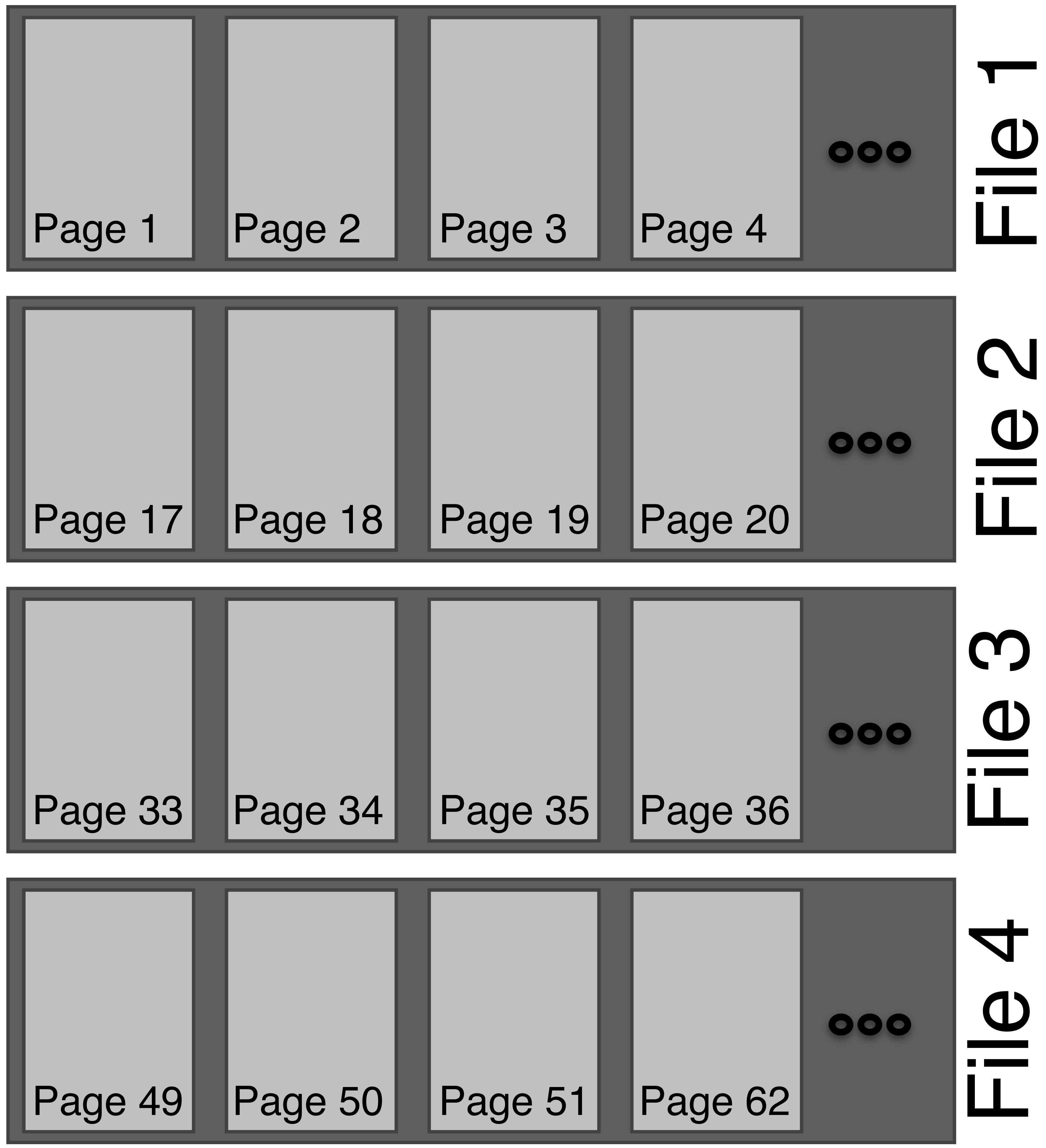
insert(Tuple 1)
insert(Tuple 2)
insert(Tuple 3)
insert(Tuple 4)



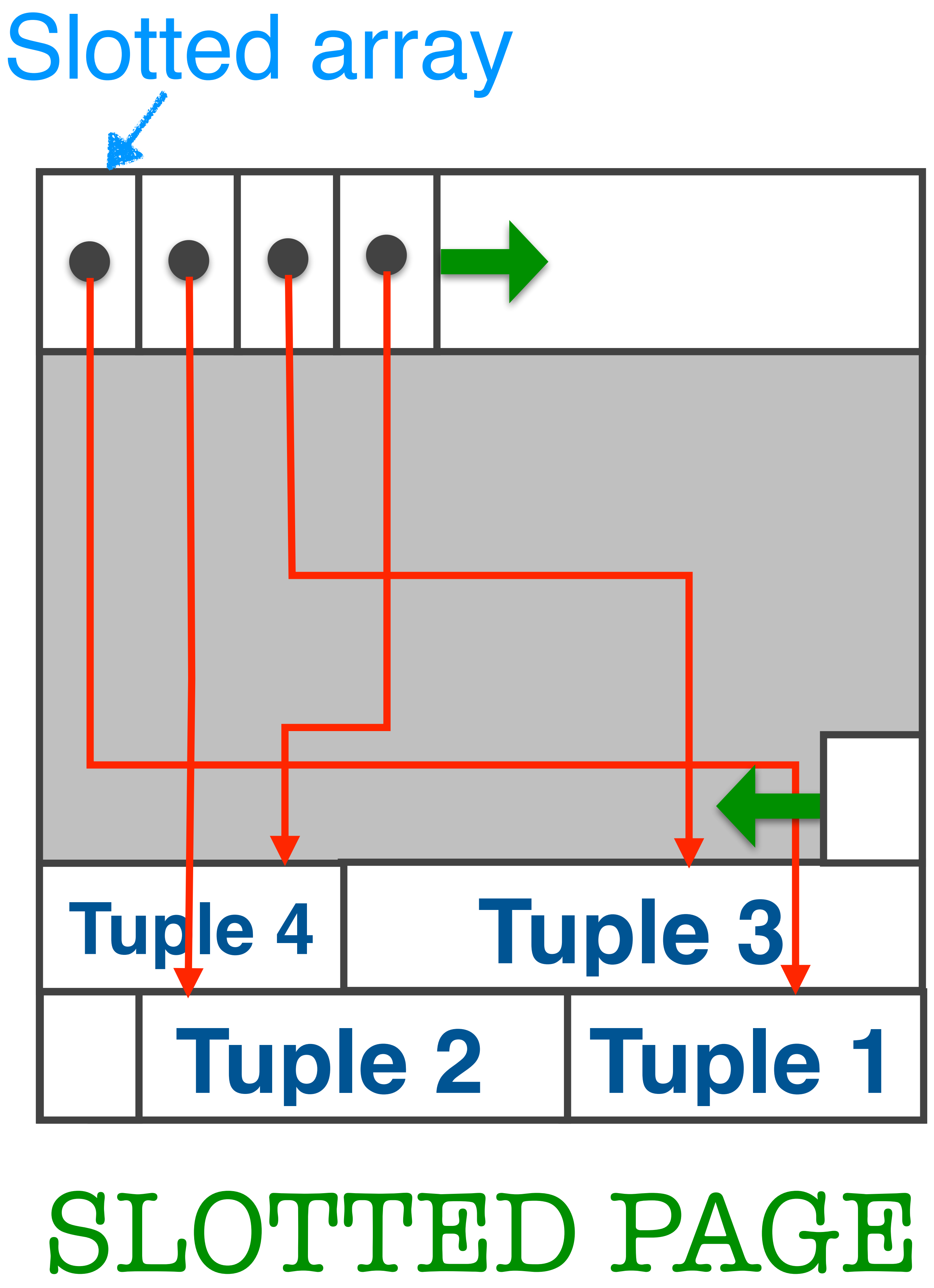
SLOTTED PAGE

Understanding **data placement**

Files, pages, tuples

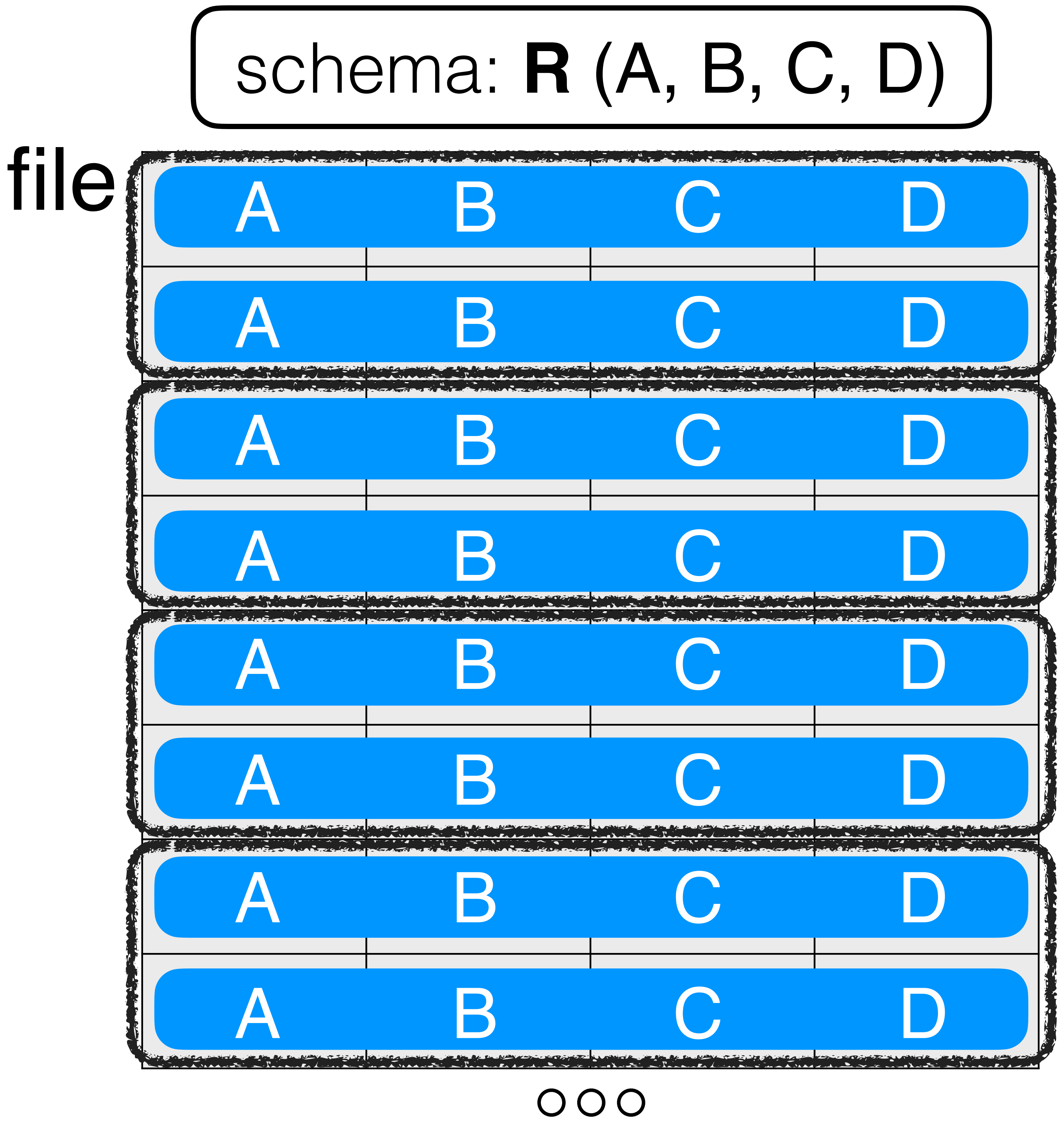


insert(Tuple 1)
insert(Tuple 2)
insert(Tuple 3)
insert(Tuple 4)



Querying over **slotted pages**

Understanding the schema



Thought Experiment 2

select A,B,C,D from R

Thought Experiment 3

select A from R

Problem?

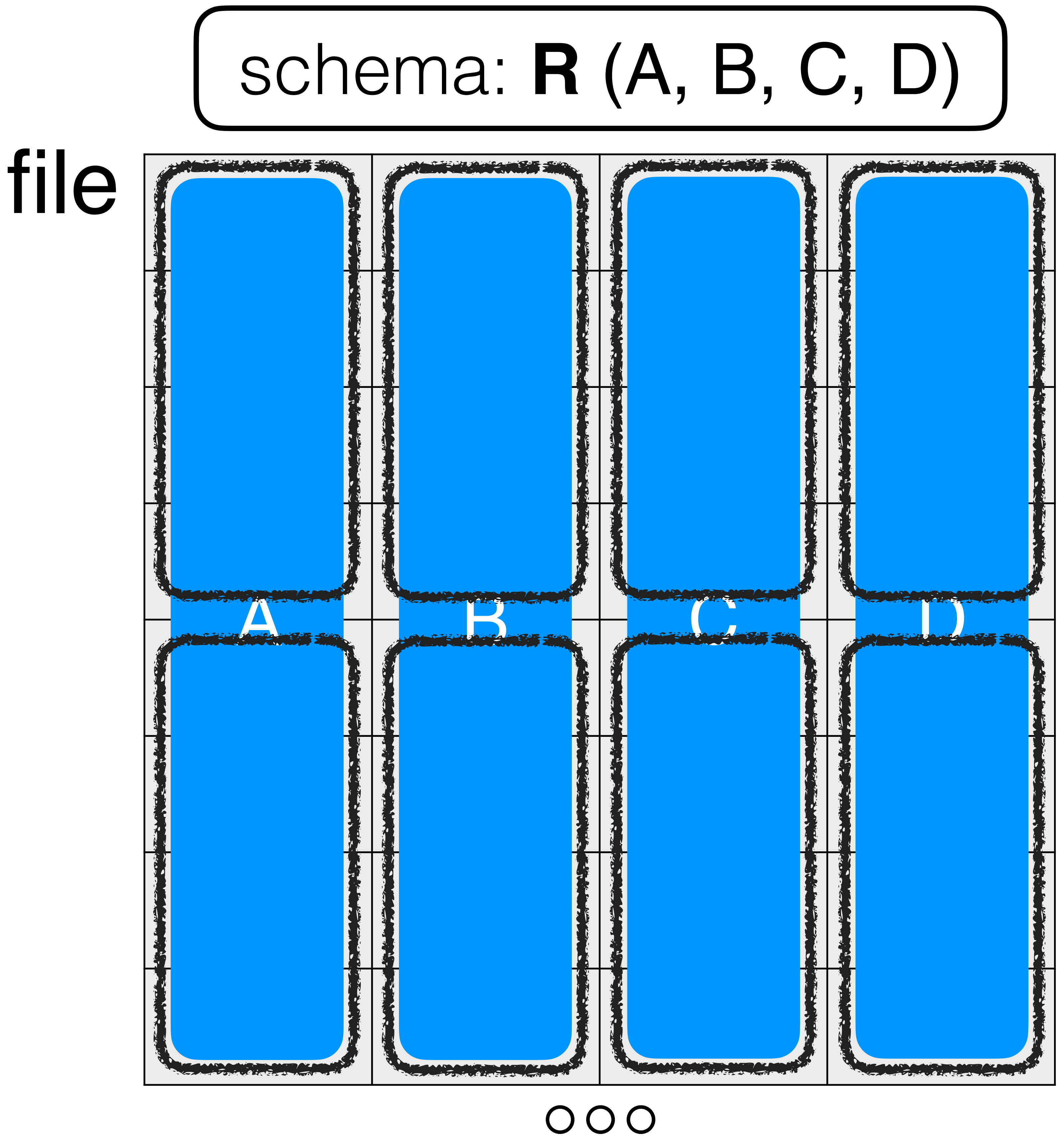
Read amplification

Solution?

Column stores

Querying over **slotted pages**

Understanding the schema



Thought Experiment 3

select A from R

Thought Experiment 2

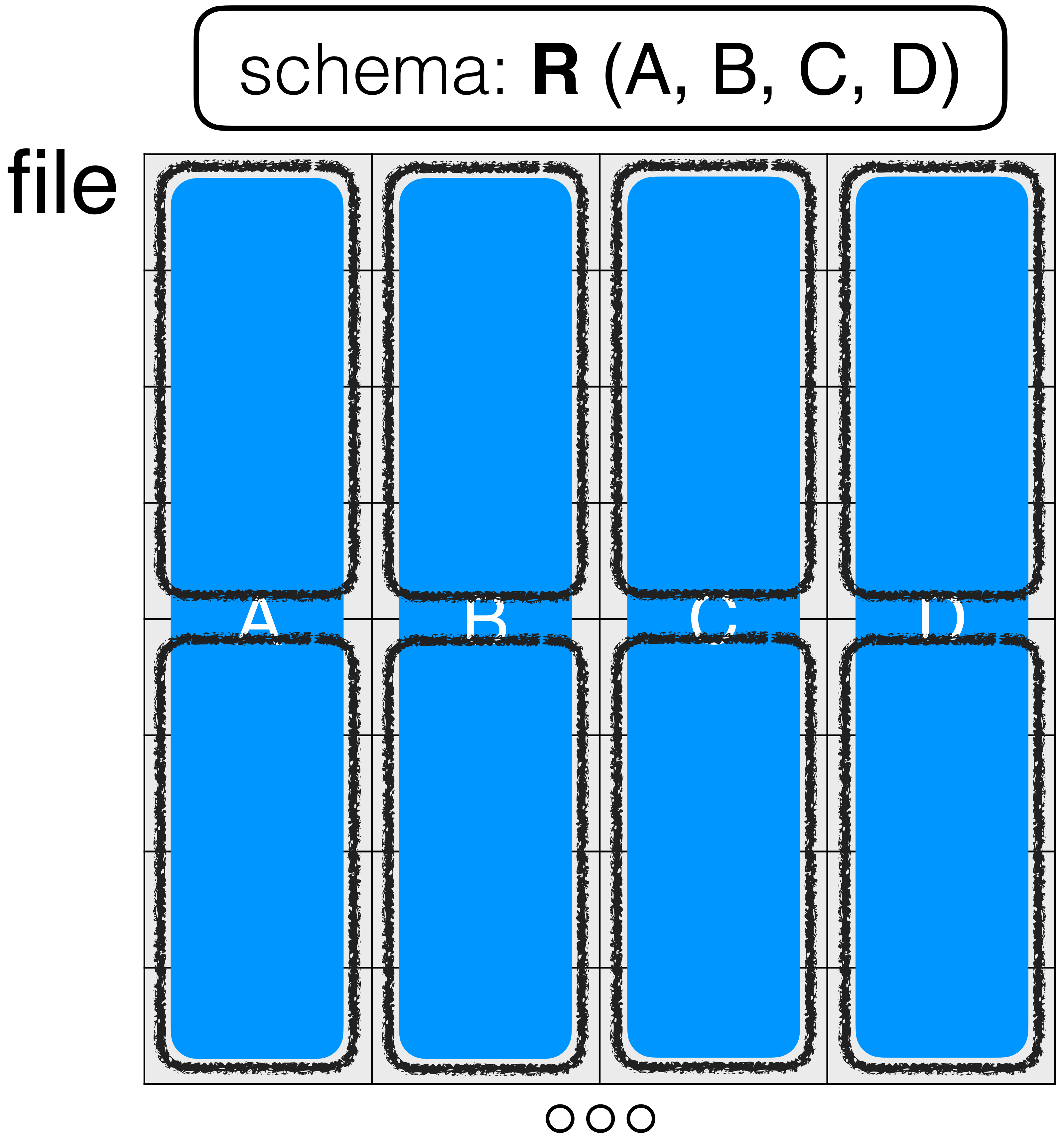
select A,B,C,D from R

Problem?

Tuple reconstruction

Querying over **slotted pages**

Understanding the schema



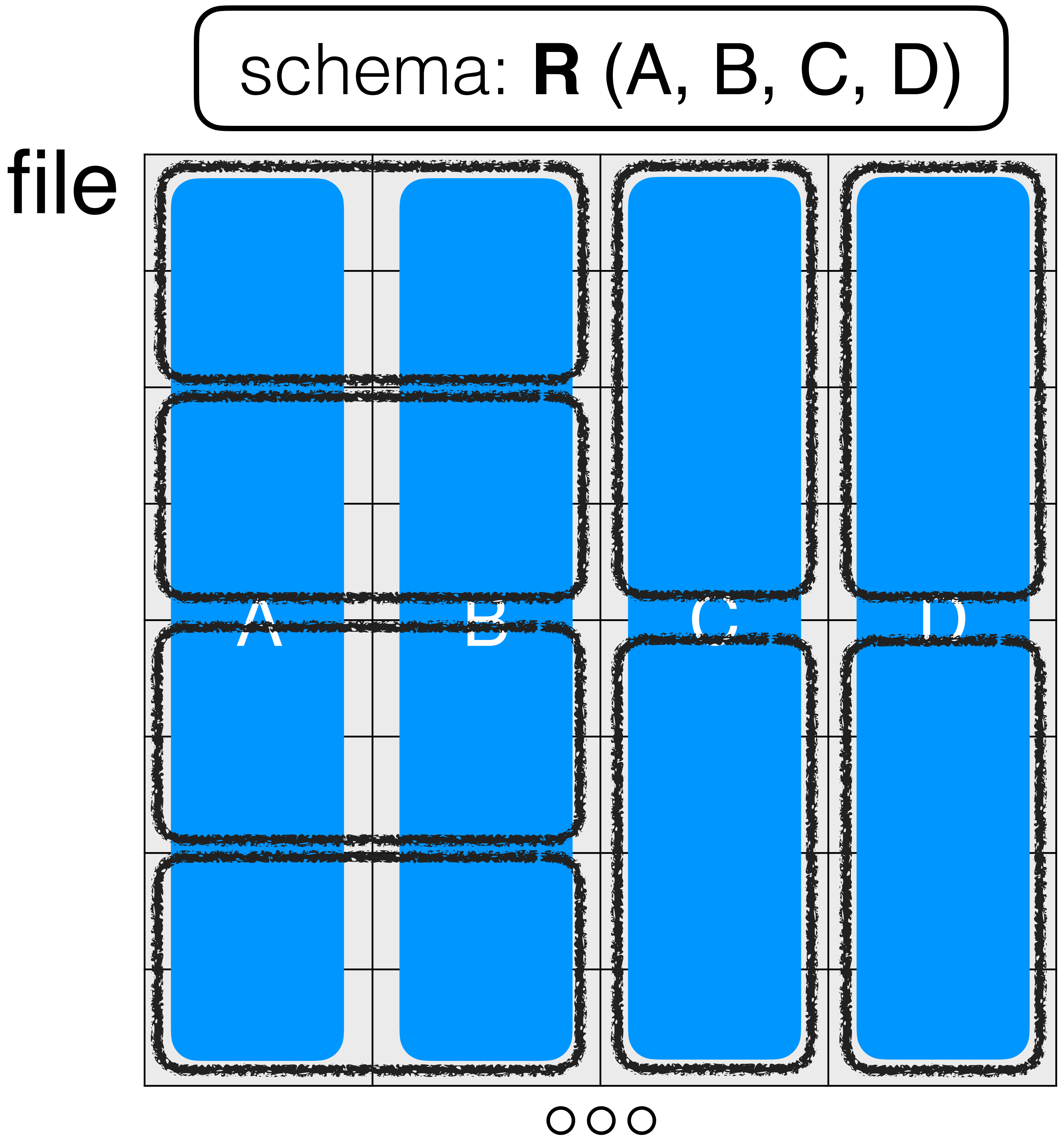
Thought Experiment 4

select A+B from R

Can we do something better?

Querying over **slotted pages**

Understanding the schema



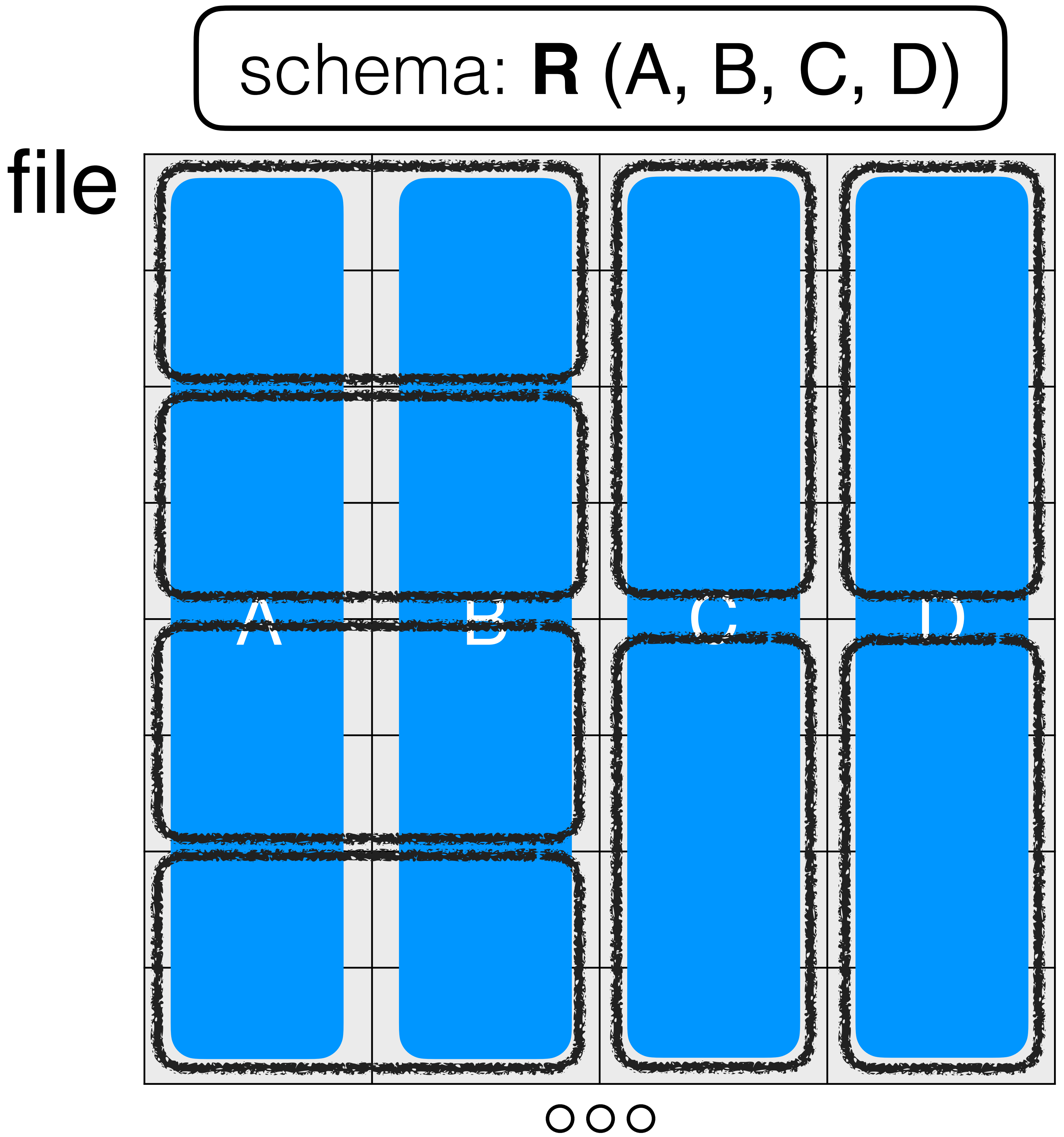
Thought Experiment 4

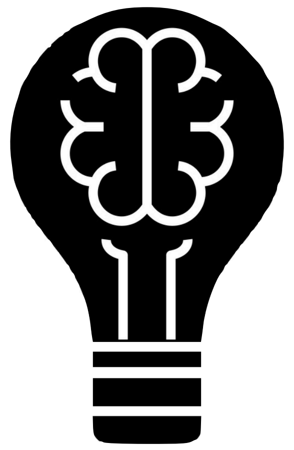
select A+B from R

Can we do something better?

Querying over **slotted pages**

Understanding the schema





Thought Experiment 5

select A+B from R
select A,B,C,D from R
select A from R

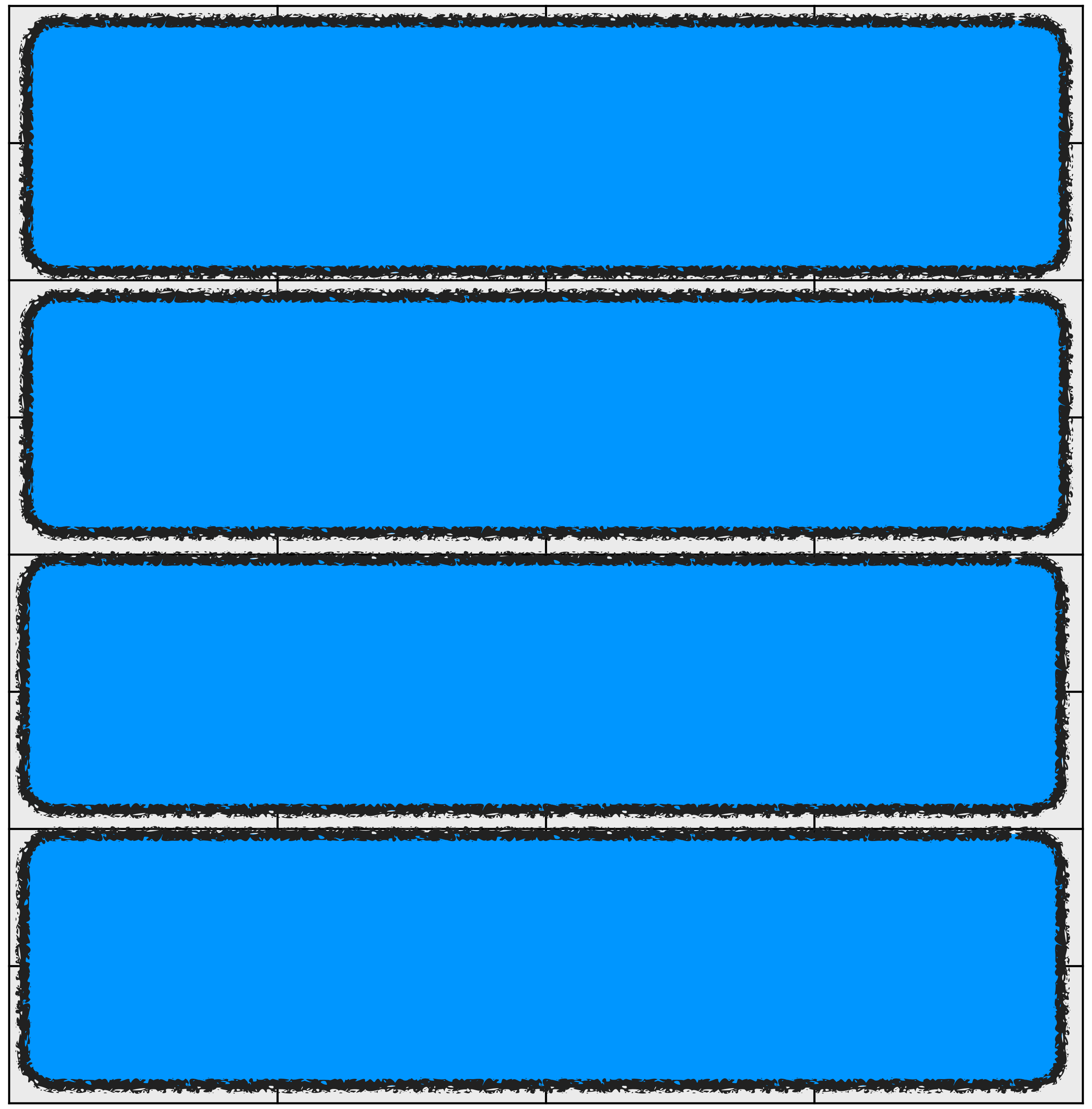
What if we have all three queries?

What if we only have inserts/updates?

Querying over **slotted pages**

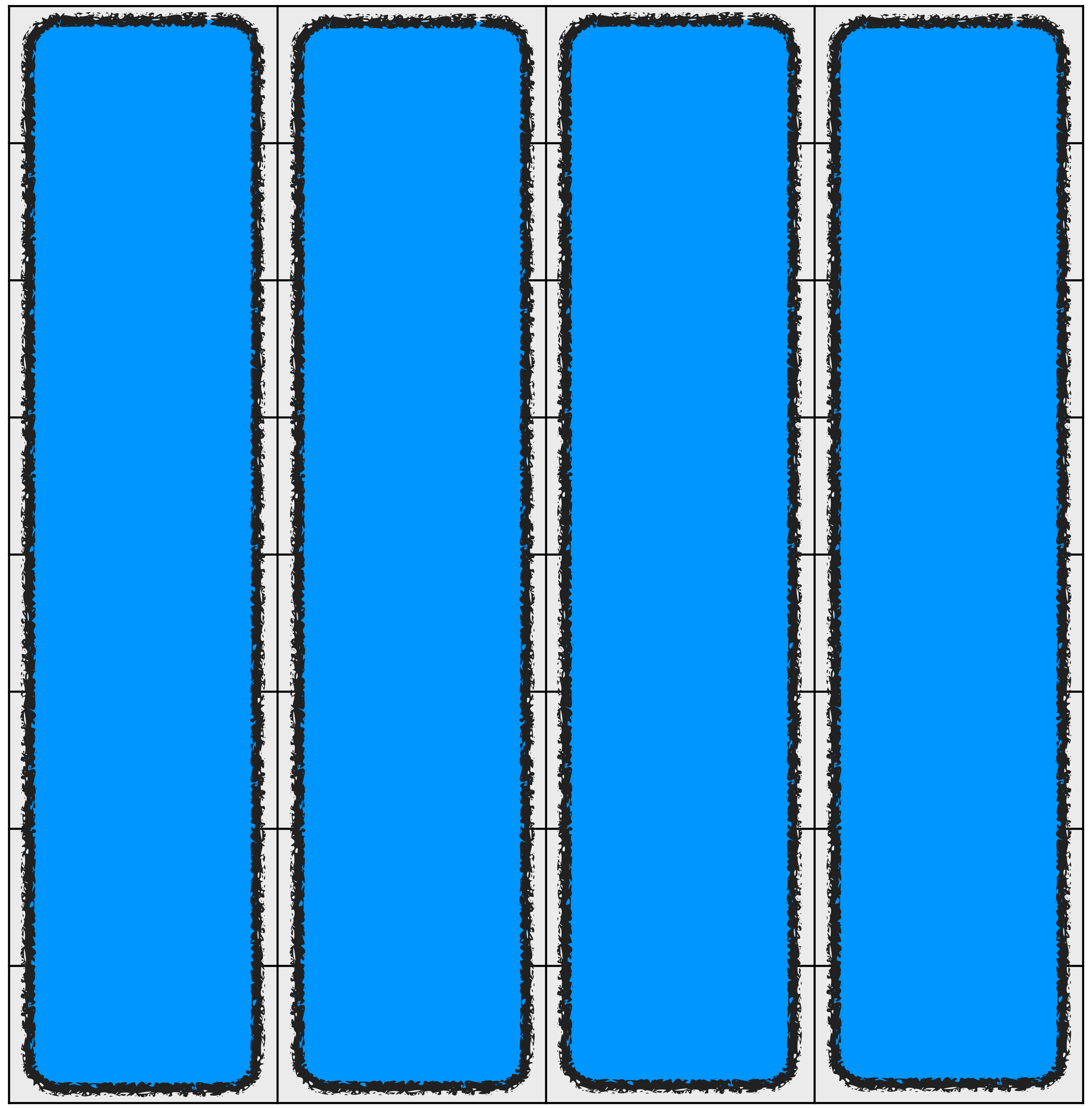
Understanding the schema

schema: **R** (A, B, C, D)



row store

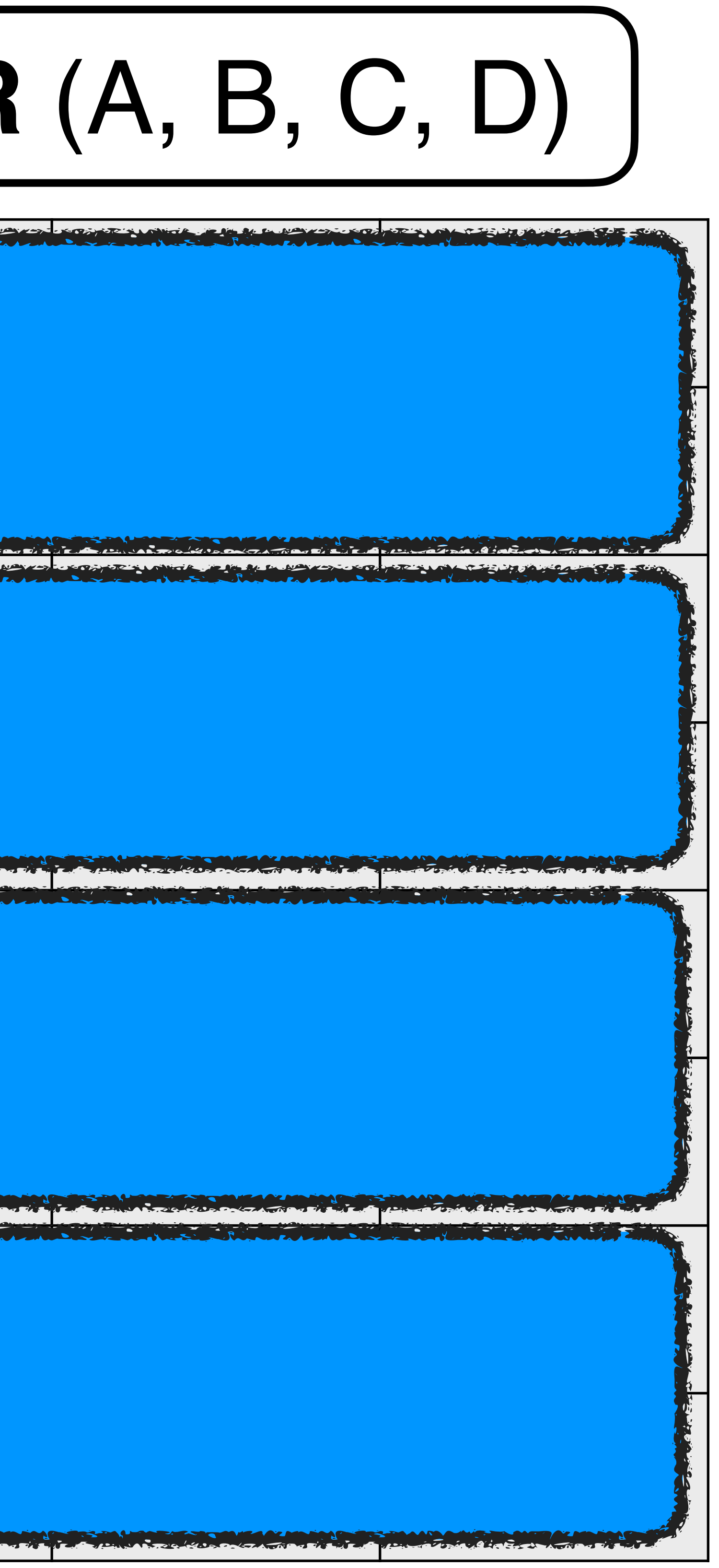
schema: **R** (A, B, C, D)



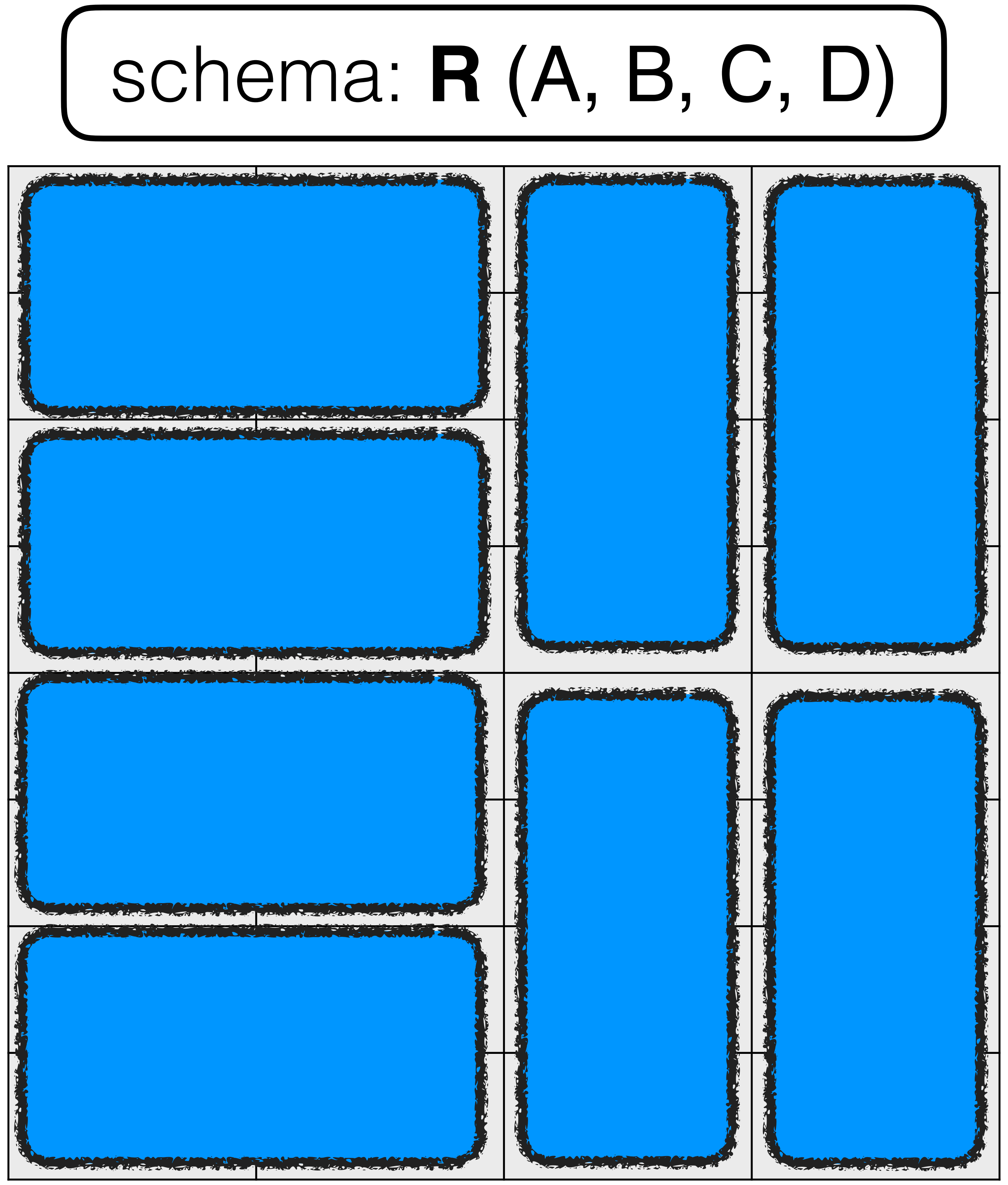
column store

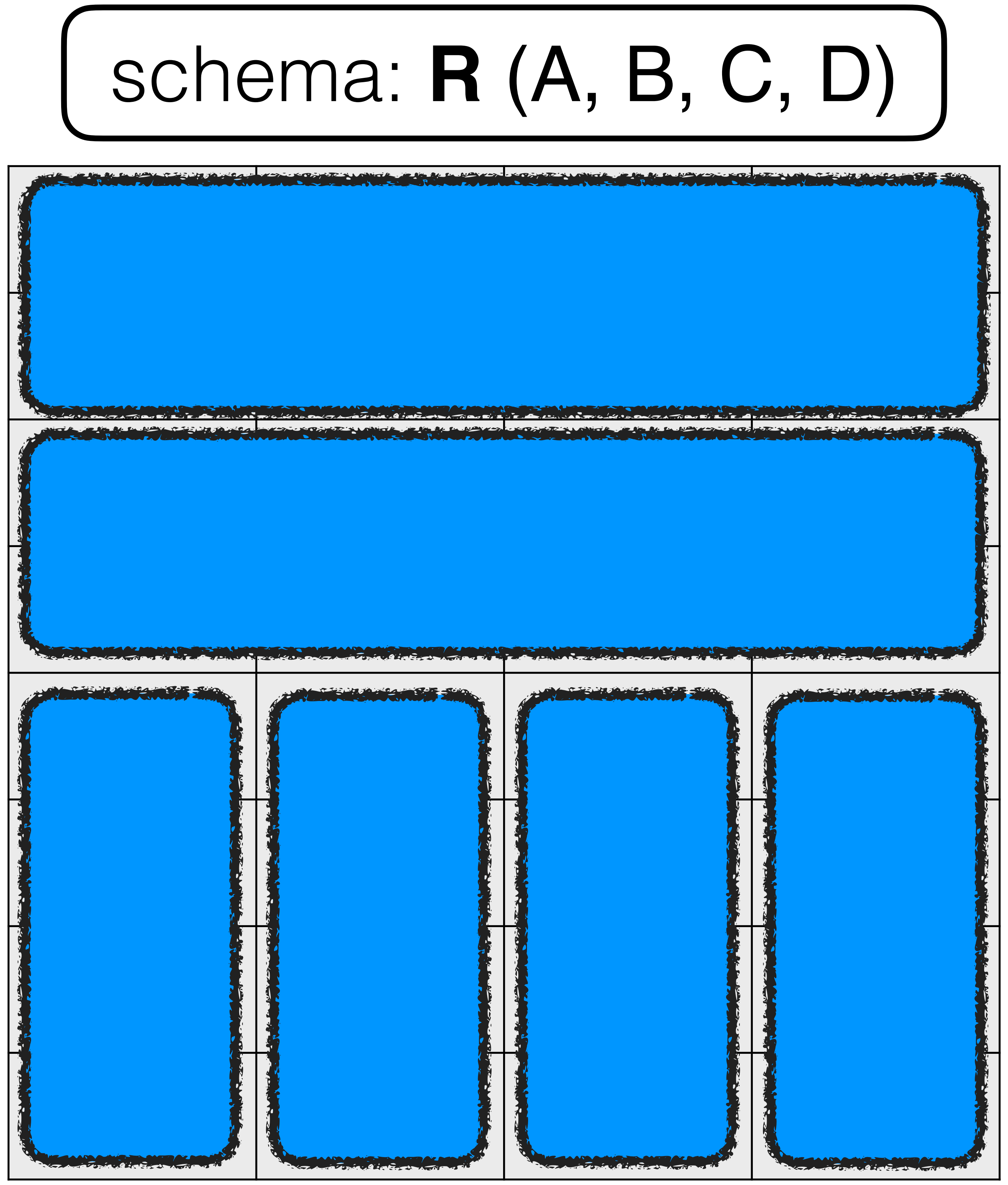
Querying over **slotted pages**

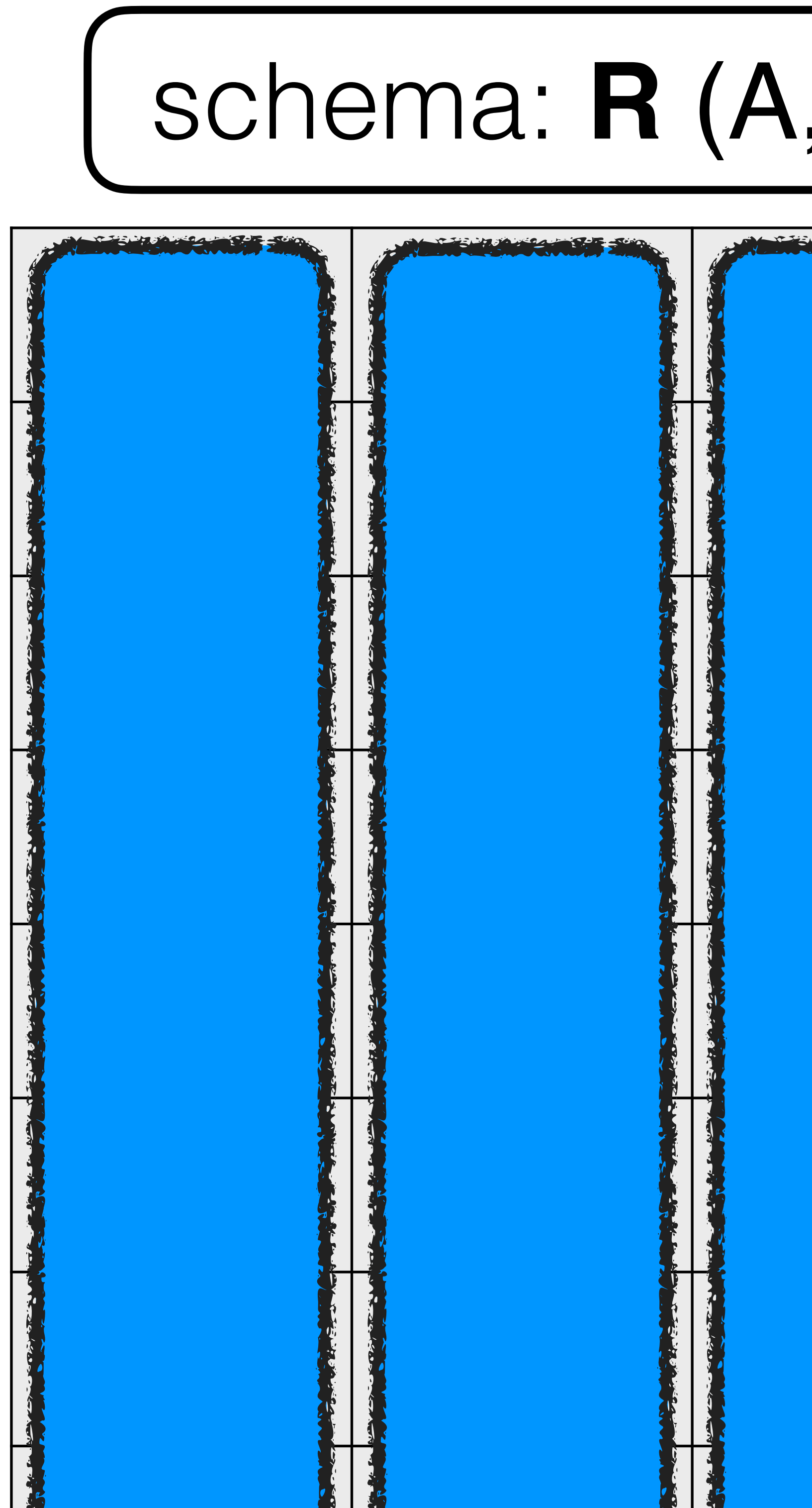
Understanding the schema



store





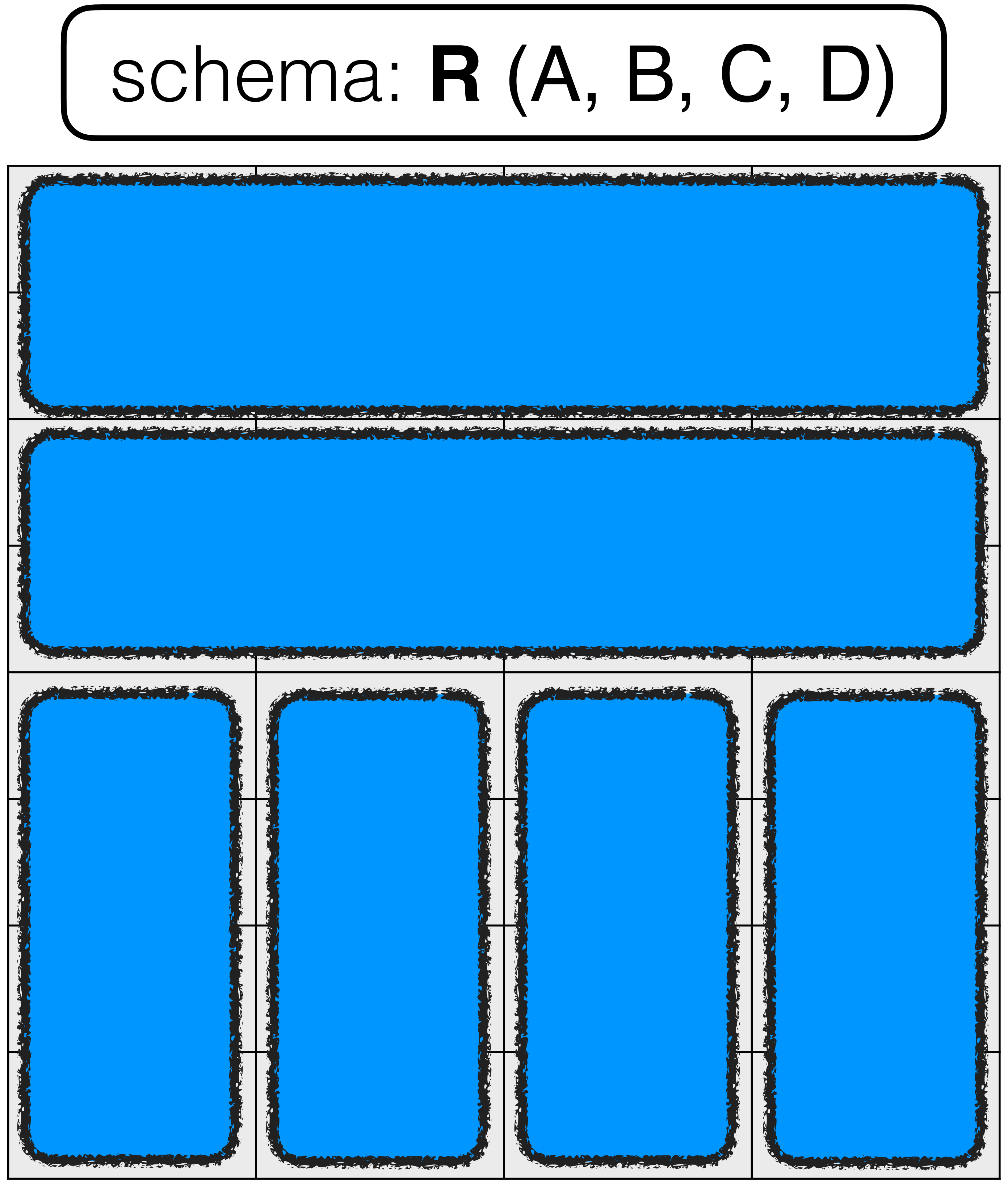


hybrid data layouts

Brandeis UNIVERSITY

Querying over **slotted pages**

Understanding the schema



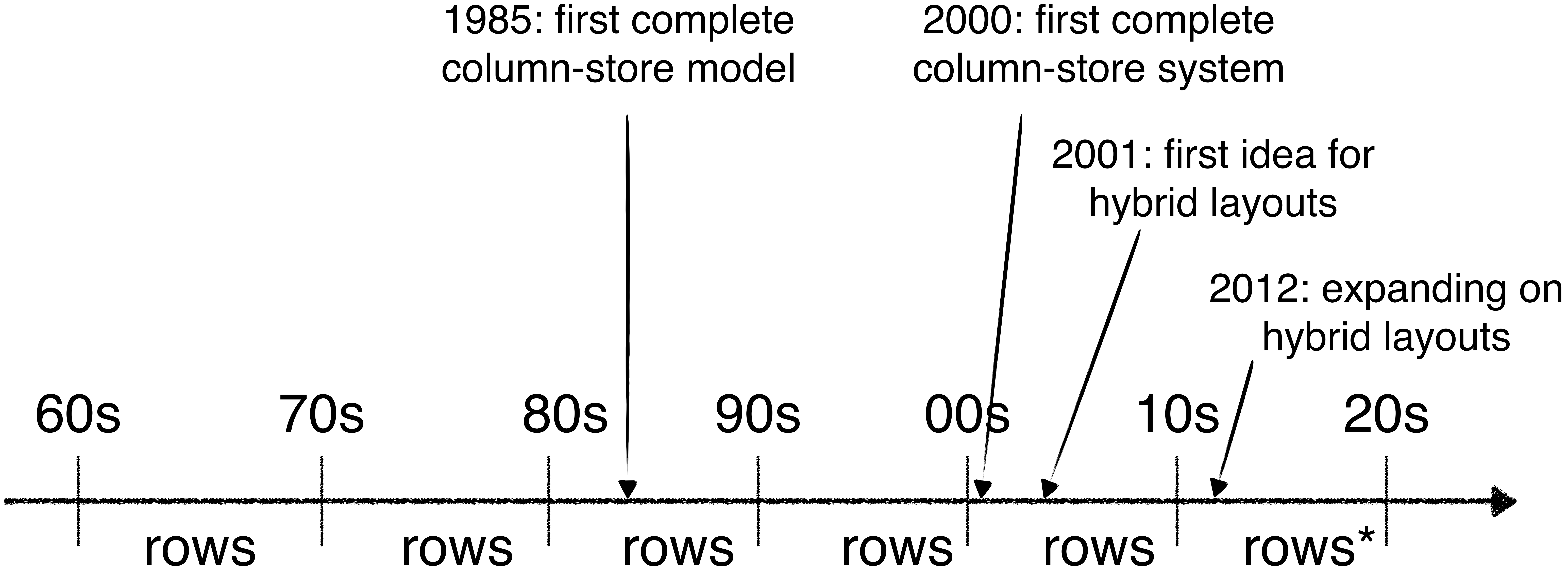
Thought Experiment 6

When would you have this data layout?

Queries on a subset of data

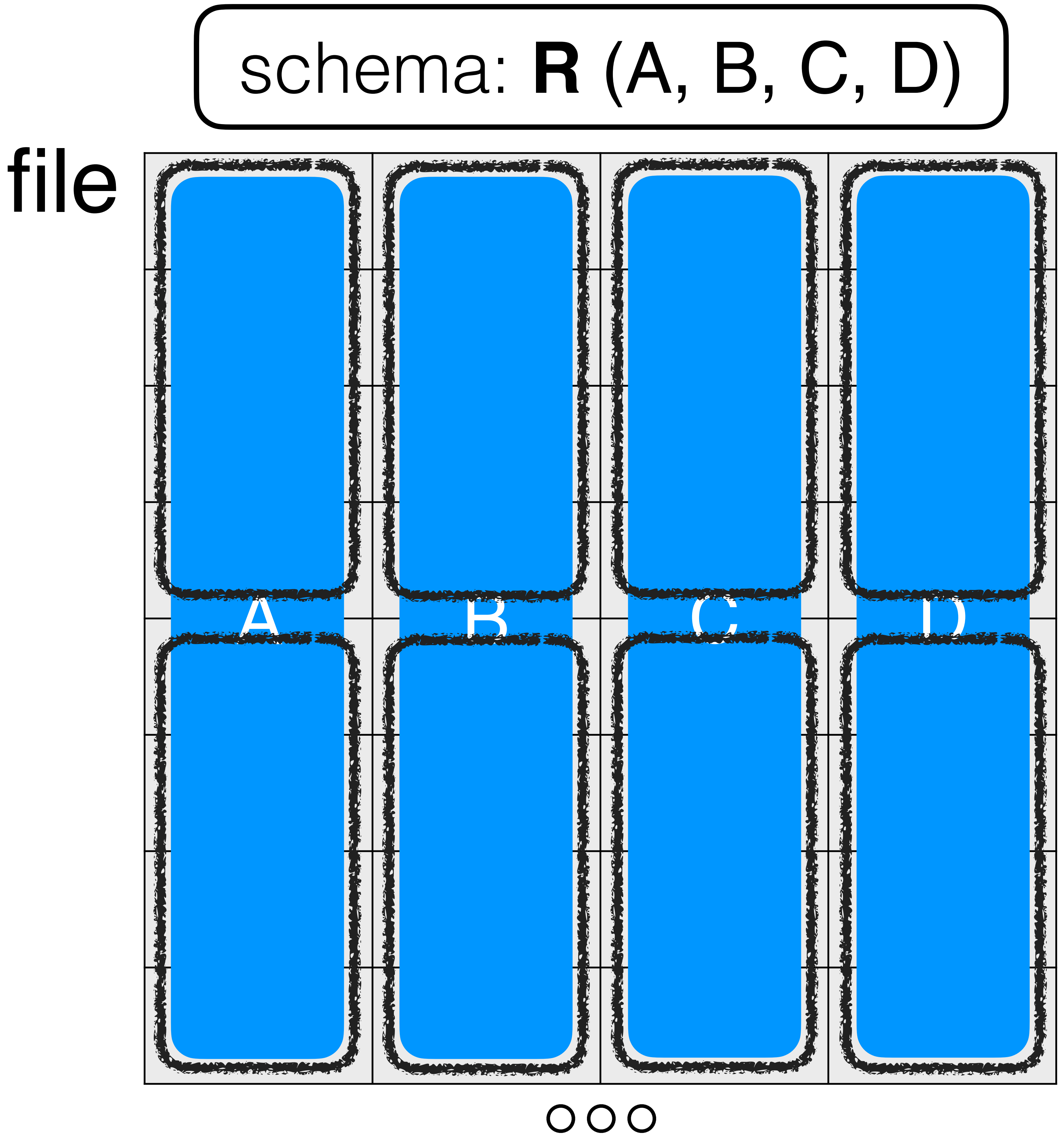
Evolution of column store

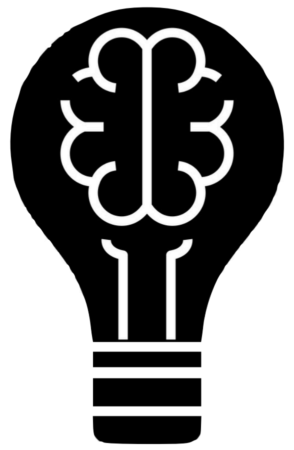
From row sotres to column stores



Querying over **slotted pages**

Understanding the schema





Thought Experiment

select max(B) from R
where A>5 and C<10

Home work!

Next time in COSI 167A

Row stores vs. Column stores

[P] "Column-Stores vs. Row-Stores: How Different Are They Really?", *SIGMOD*, 2008

TECHNICAL QUESTION 1

[B] "C-Store: A Column-oriented DBMS", *VLDB*, 2005

Brandeis UNIVERSITY

COSI 167A

Advanced Data Systems

Class 3

Data Systems Architecture

Prof. Subhadeep Sarkar

https://ssd-brandeis.github.io/COSI-167A/

Brandeis UNIVERSITY