CSCD443 - Database Systems Technology

Winter 2023
Niv Dayan

We will start at 2:10 pm

Who am I?

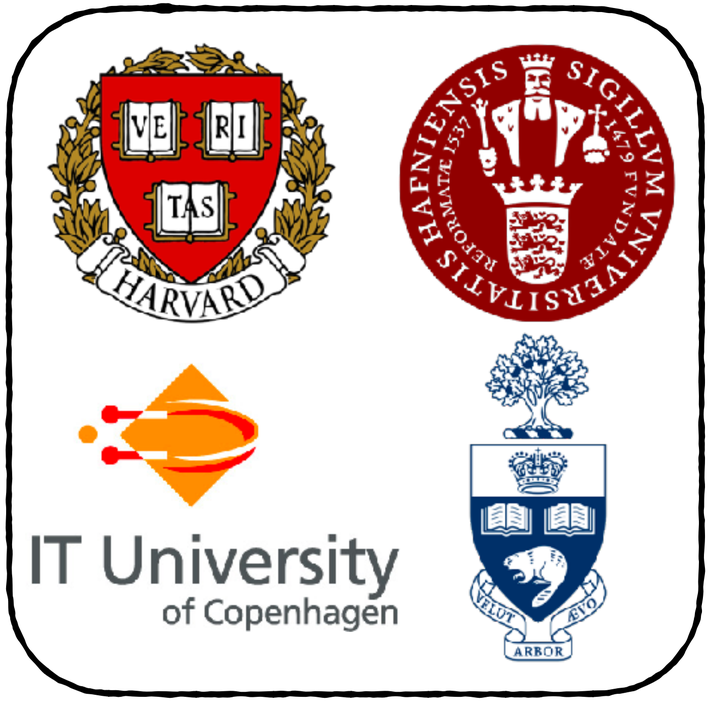
>10 years of research experience in data structures & algorithms for databases

https://www.nivdayan.net/

Who am I?

10 years of research experience in data structures & algorithms for databases

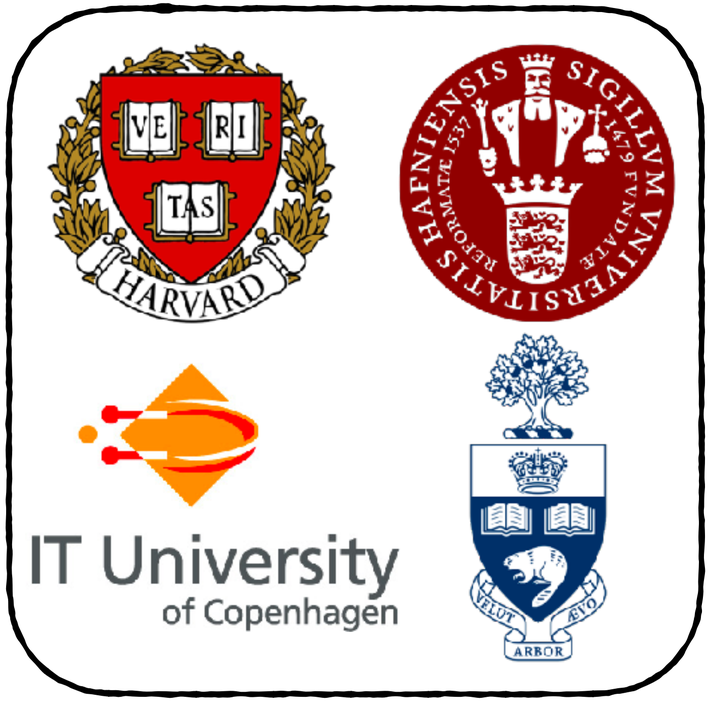
In academia



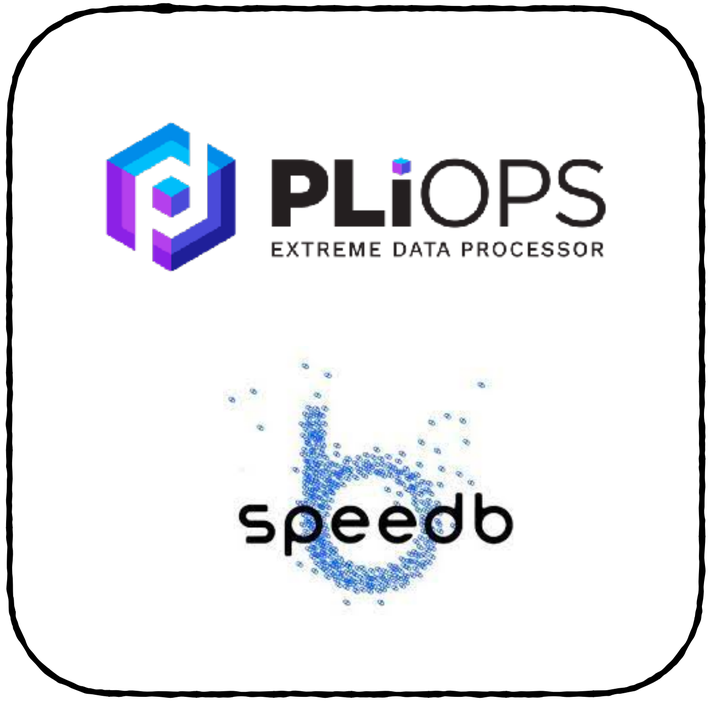
Who am I?

10 years of research experience in data structures and algorithms for databases

In academia



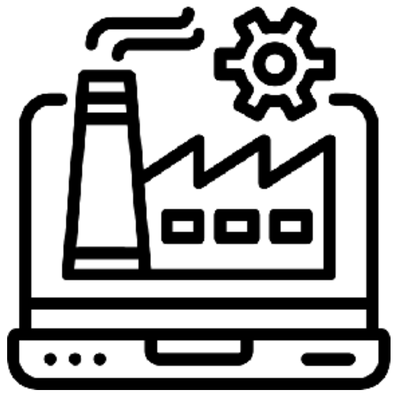
And in industry



This course combines both

Theory

Practice



New at UofT :)

Visa delays

Who are you? Prerequisites…

Who are you? Prerequisites…

Introduction to Databases

Relational algebra, data modeling, SQL

Who are you? Prerequisites…

Introduction to Databases

Relational algebra, data modeling, SQL

Operating Systems

Concurrency & synchronization
File systems, virtual memory

Who are you? Prerequisites…

Introduction to Databases

Relational algebra, data modeling, SQL

Operating Systems

Concurrency & synchronization
File systems, virtual memory

Design and Analysis of Data Structures

Binary trees, sorting, hash tables, priority queues, Big-O analysis

Who are you? Prerequisites…

Introduction to Databases

Relational algebra, data modeling, SQL

Operating Systems

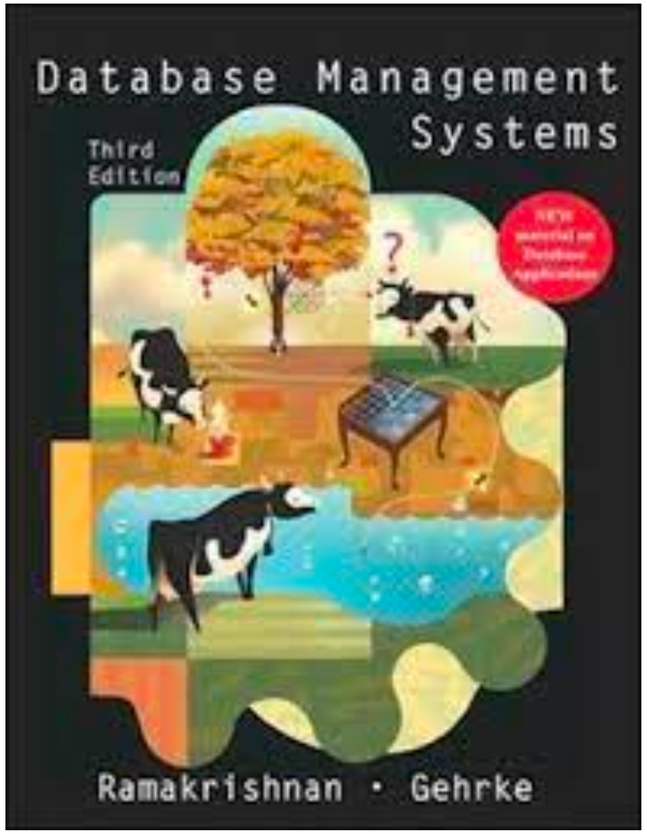
Concurrency & synchronization
File systems, virtual memory

Design and Analysis of Data Structures

Binary trees, sorting, hash tables, priority queues, Big-O analysis

Solid programming skills in C, C++, Java, or at least Python

Reading

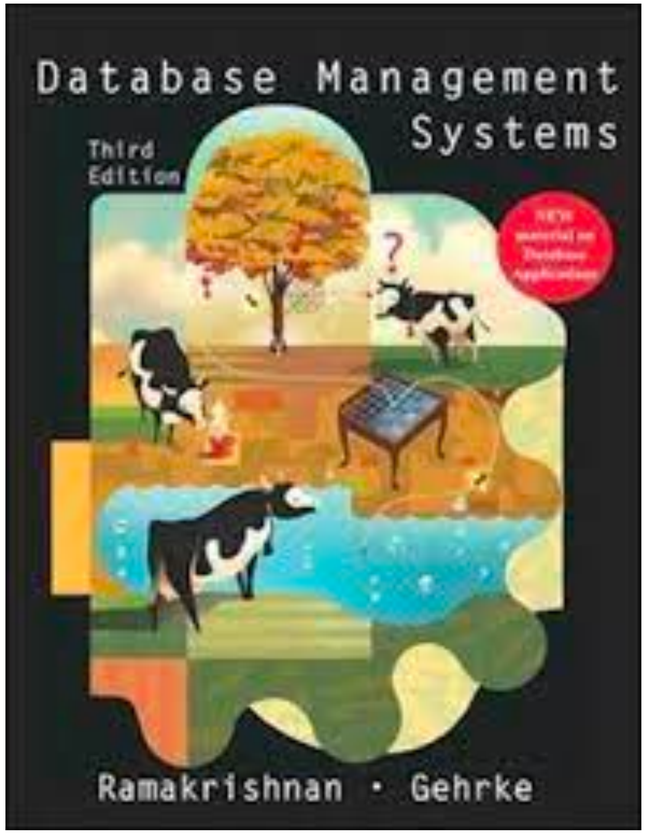


"Database Management Systems” 3rd Edition
Raghu Ramakrishnan & Johannes Gehrke

Classic, but dated (20 years old). A lot has changed

We’ll use it as a base, along with the slides and supplementary material to reflect 20 years of progress

Reading



Part 1 - Introduction	(Chapter 1)
Part 3 - Storage & Indexing	(Chapters 8-11)
Part 4 - Query Evaluation	(Chapters 12-15)
Part 5 - Transaction Management	(Chapters 16-18)

Check the course website for what to read each week

01 Week 1: Introduction & Storage Devices

After introducing the course and the high-level architecture of database systems, we will take a deep look at the memory hierarchy. As we will see throughout the course, the properties of the memory hierarchy give rise to how modern databases are architected.

For an introduction to the course, first read Chapter 1 in the textbook. For information on storage, read Chapter 8 Part 8.1, and Chapter 9 Parts 9.1 and 9.2.

As you will notice, the textbook was written before SSDs became popular. Therefore, have a look at the following article for background on SSDs.

https://flashdba.com/2014/09/17/understanding-flash-the-flash-translation-layer/

I also invite those who are interested to look at one of my earliest research papers on FTL design for SSDs. Parts 1 and 2 would suffice, but you are welcome to read further.

https://nivdayan.github.io/GeckoFTLPaper.pdf.

02 Week 2: Table and Buffer Management

03 Week 3: Column-Stores

Questions are always welcome!

Two-hour lectures can be long. Let’s make it interesting.

Problem-based learning

Lectures are embedded with thinking exercises

Please participate!

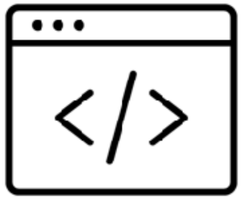
Post questions for everyone’s benefit!

Group Project

Mini-database system

More on Thursday

C++



Groups of 3

From into Groups

Send an email by this time next week if you do not already have a group, and I will pair you up.

Marking Scheme

Project (40%)

Midterm (20%)

Final (40%)

TAs

Akshay Bapat

Pooyan Habibi

Tutorials

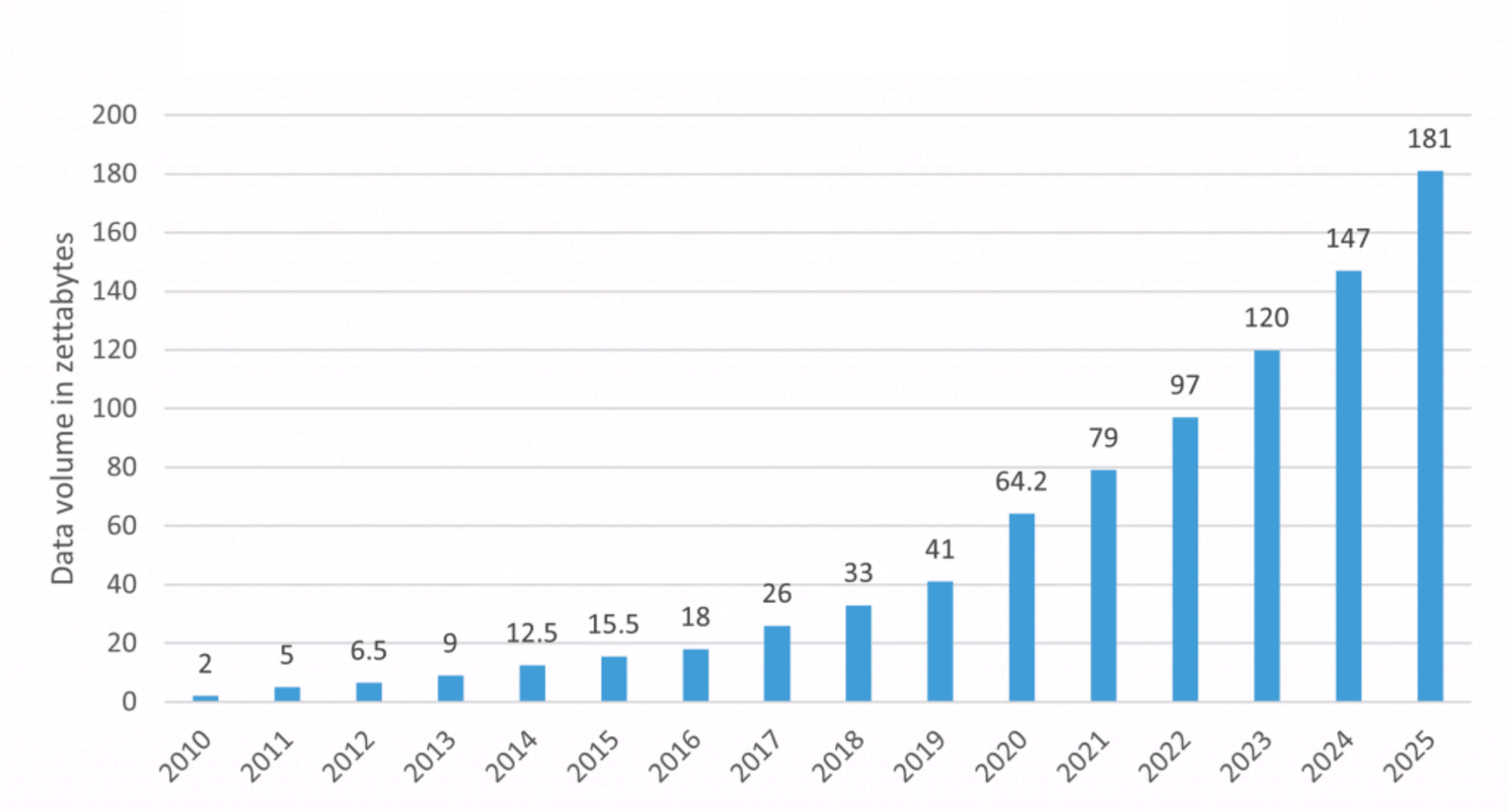
Exercises

Project

Research

Lectures

Every 2 years, the amount of data humanity produces doubles



	# bytes
Zettabytes	2^{70}
Exabytes	2^{60}
Petabytes	2^{50}
Terabytes	2^{40}
Gigabytes	2^{30}
Megabytes	2^{20}
Kilobytes	2^{10}

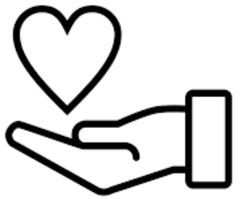
This data is…

stored by data management systems.

This data is…

stored by data management systems.

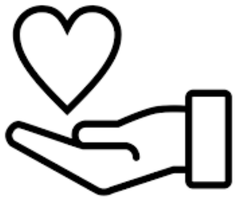
used everywhere
(business, science, NGOs, etc)



This data is…

stored by data management systems.

used everywhere
(business, science, NGOs, etc)

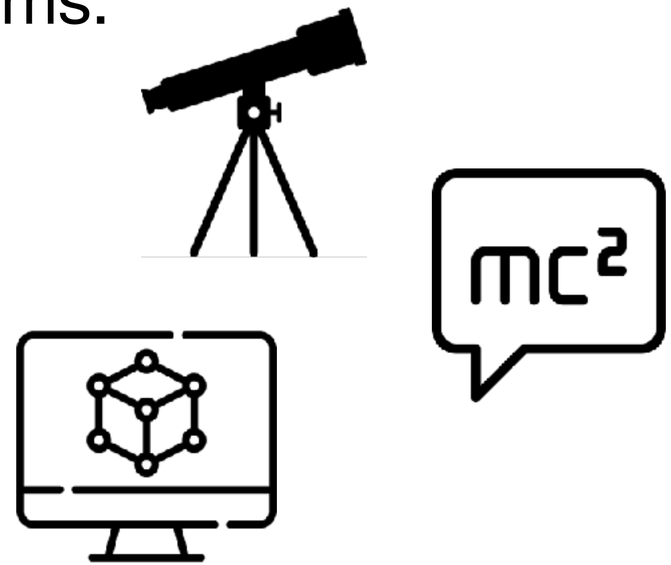


You will likely work with data, no matter where you go

The fourth paradigm

Scientific discovery has moved along three paradigms:

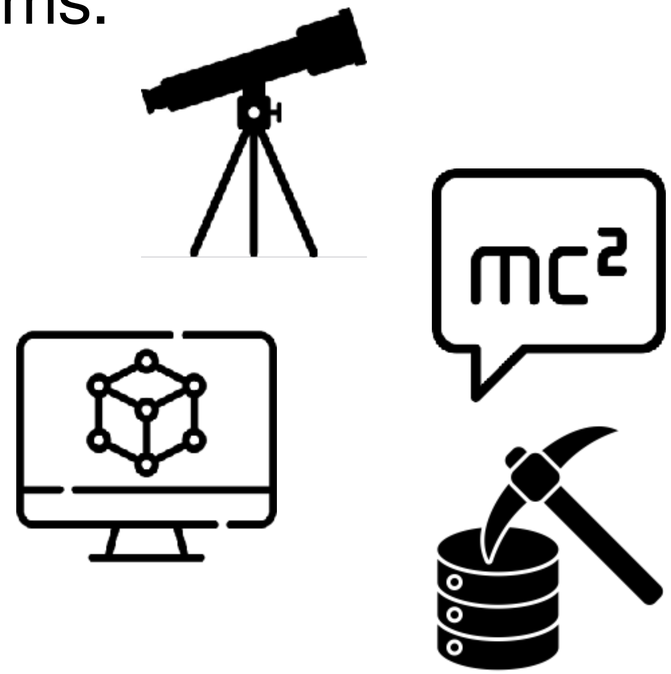
1. Observation & Experimentation
2. Theoretical approaches
3. Computational science & simulation



The fourth paradigm

Scientific discovery has moved along three paradigms:

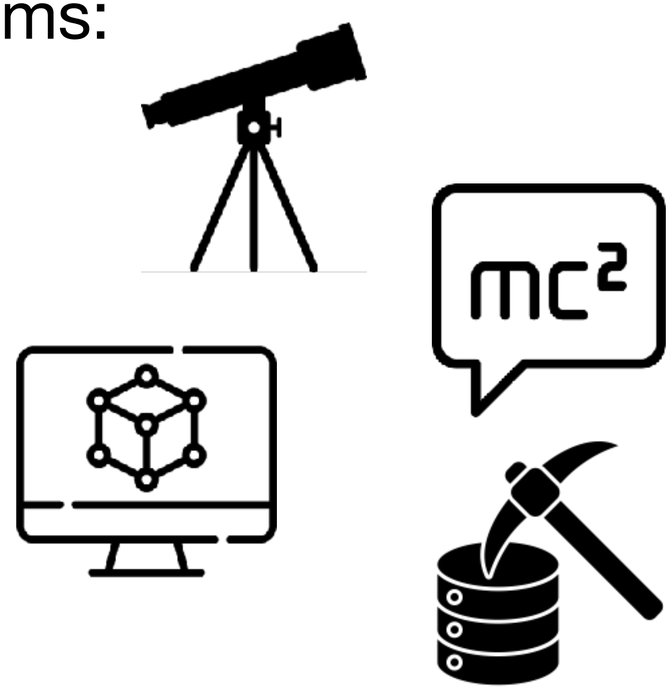
1. Observation & Experimentation
2. Theoretical approaches
3. Computational science & simulation
4. **Data-driven discovery**



The fourth paradigm

Scientific discovery has moved along three paradigms:

1. Observation & Experimentation
2. Theoretical approaches
3. Computational science & simulation
4. Data-driven discovery



“The data for the next big discovery may already be available. But it needs to be found and analyzed.”

Big Data

Admittedly, a buzzword

Big Data

Admittedly, a buzzword

Refers to growth of data beyond our ability to curate and derive meaning from it. Since early 2000’s.

Big Data

Admittedly, a buzzword

Refers to growth of data beyond our ability to curate and derive meaning from it. Since early 2000's.

"One size does not fit all."

A (rough) timeline

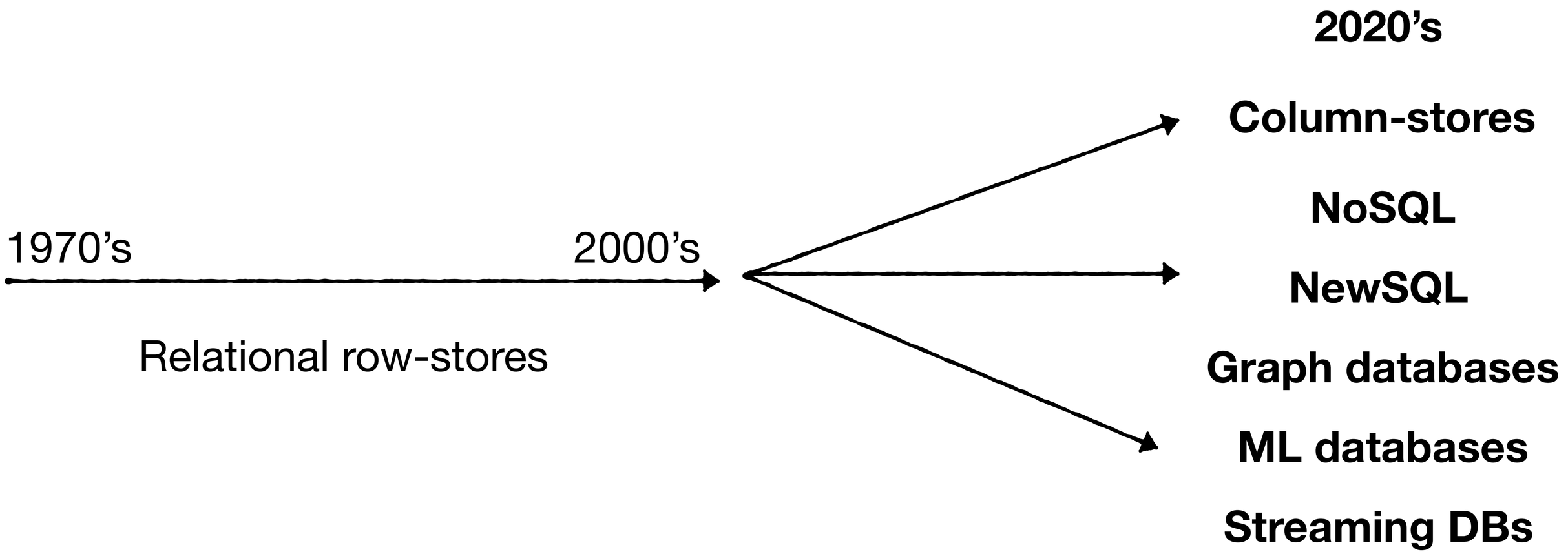
Relational row-stores

A (rough) timeline

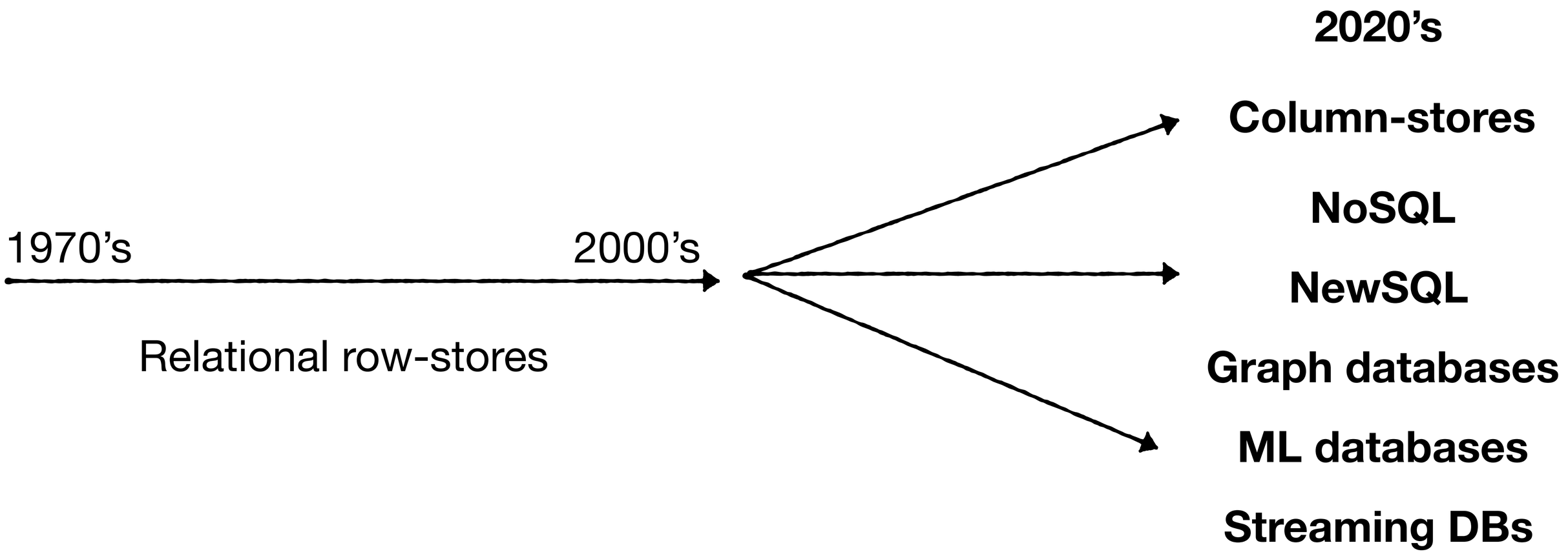
1970’s 2000’s

Relational row-stores

A (rough) timeline



A (rough) timeline

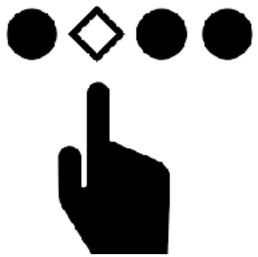


We can’t cover all of them, but we’ll give you basic design principles underpinning them all.

What will this course help you with?

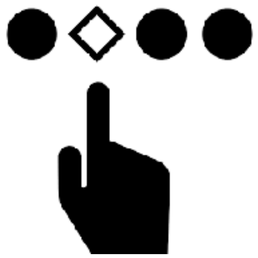
What will this course help you with?

Choose



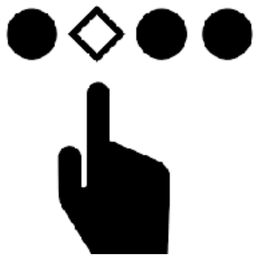
How to choose a database for a particular use-case.

Choose **Use**



Model, index & query the data with performance in mind

Choose

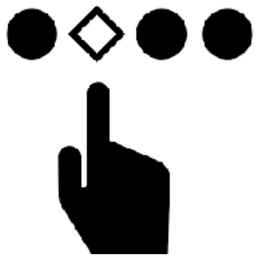


Use

Tune

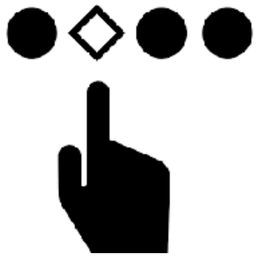
Tune parameters to improve performance

Choose Use Tune **Extend**



Add functionality and improved algorithms

Choose



Use

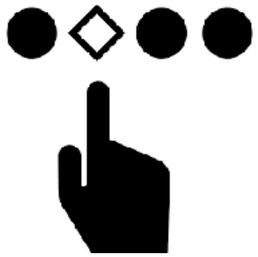
Tune

Extend

Build

Build new databases for new use-cases
(ML, self-driving cars, VR, etc)

Choose



Use

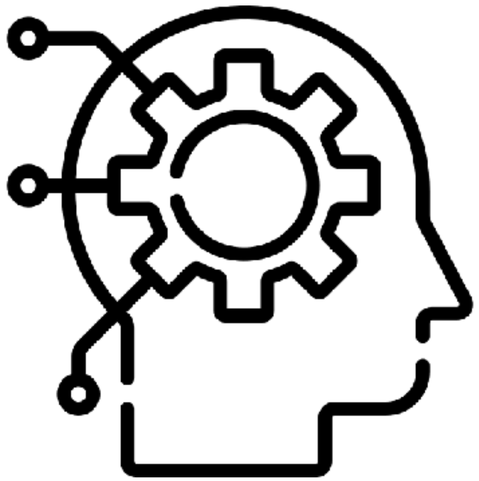
Tune

Extend

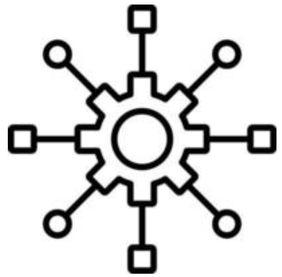
Build

These all require deep knowledge of database anatomy

Themes & learning outcomes

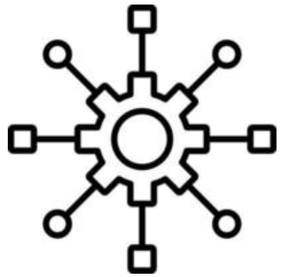


Algorithms & data structures

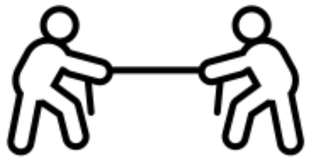


The nuts and bolts of every database system

Algorithms &
data structures

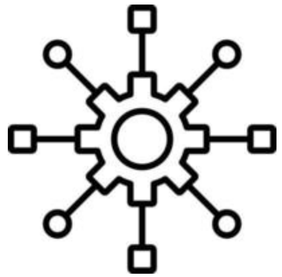


Trade-offs

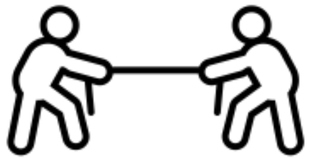


There is usually many design options with different trade-offs.

Algorithms &
data structures



Trade-offs



Performance

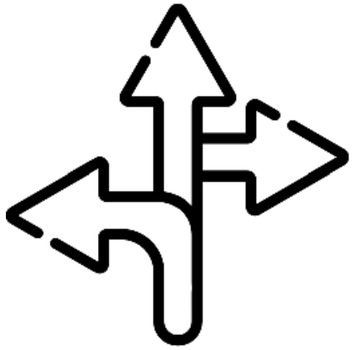
Do more with less

Get involved in research

Get involved in research

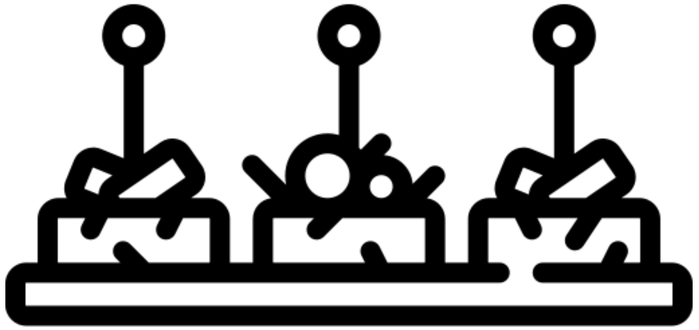
Young field

Lots of directions



Lots to discover

Appetizers Menu



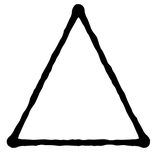
Appetizers Menu

Storage

Tables

Buffer Management

Indexes



Sorting

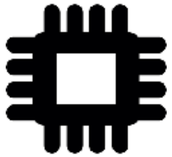
Operators

Query Optimization

Transactions

Recovery

Storage



CPU caches

memory

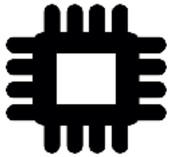
SSD

disk

Expensive & fast

Slow & cheap

Storage



CPU caches

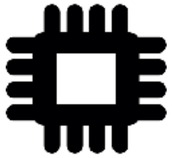
memory

SSD

disk

Optimize data movement

Storage



CPU caches

memory

SSD

disk

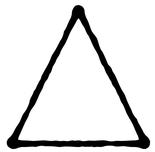
The architecture of modern DBs emerges from the properties of the memory hierarchy

Storage

Tables

Buffer Management

Indexes



Sorting

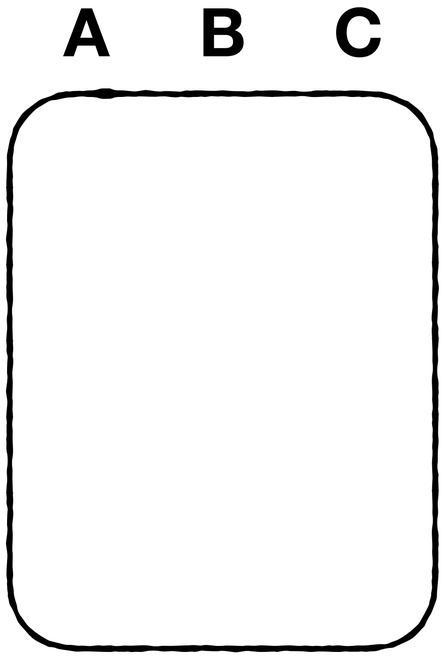
Operators

Query Optimization

Transactions

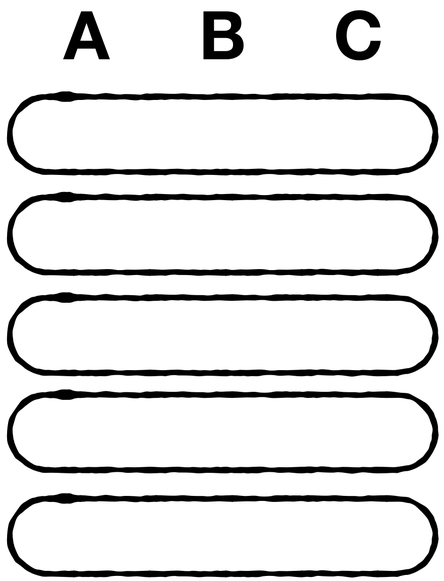
Recovery

Tables

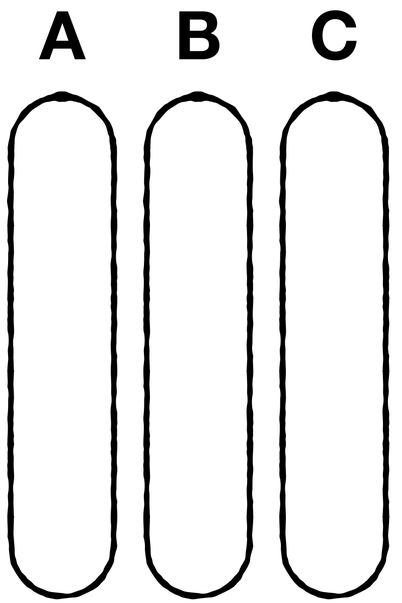


efficient inserts, updates, deletes & scans?

Tables



Row-Stores



Column-Stores

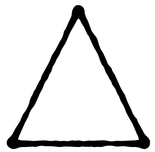
Structuring tables to optimize for different types of queries

Storage

Tables

Buffer Management

Indexes



Sorting

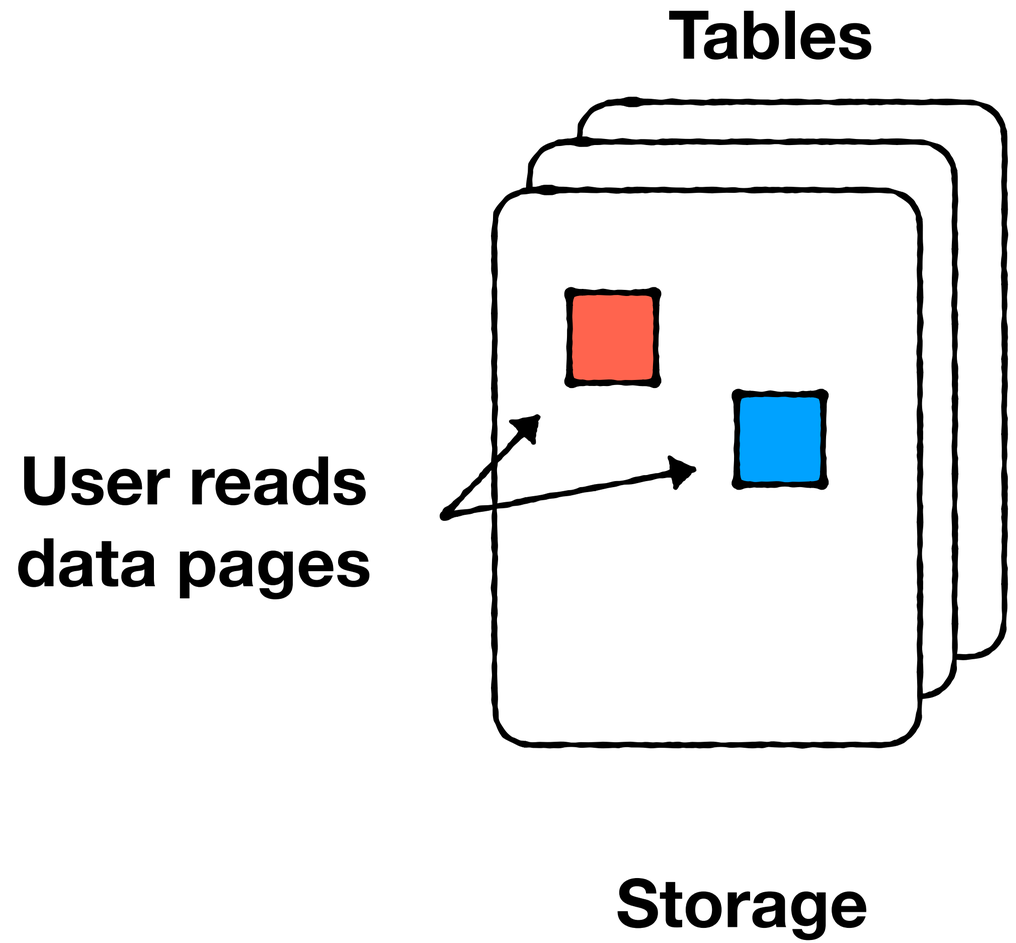
Operators

Query Optimization

Transactions

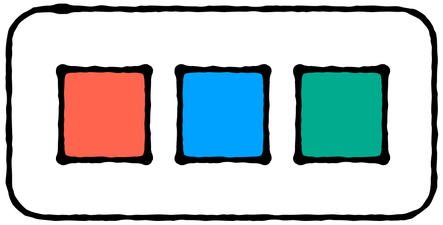
Recovery

Buffer Management

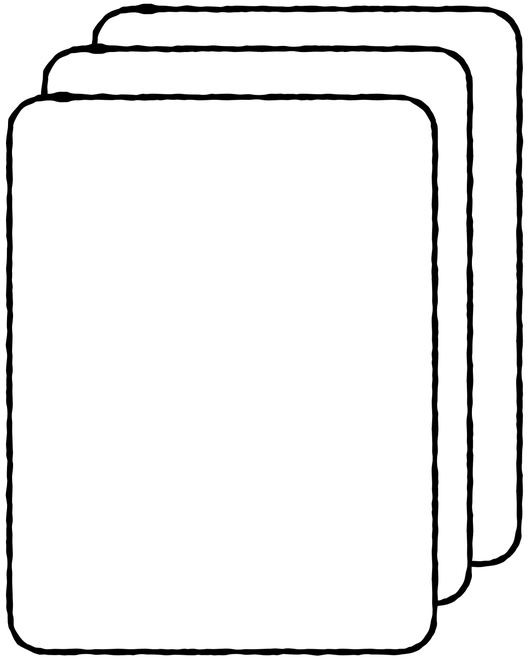


Buffer Management

Buffer Pool



Tables

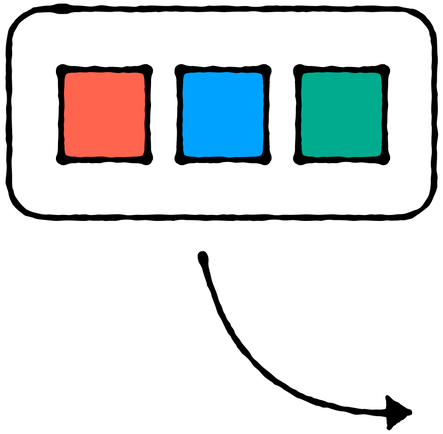


Memory

Storage

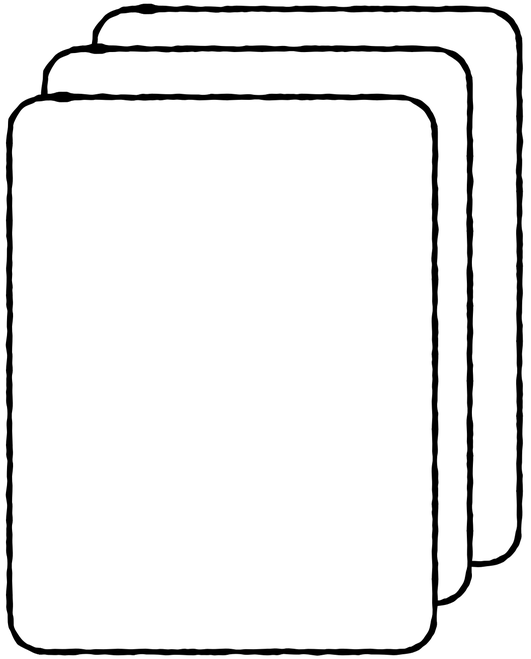
Buffer Management

Buffer Pool



Which page to evict when we run out of space?

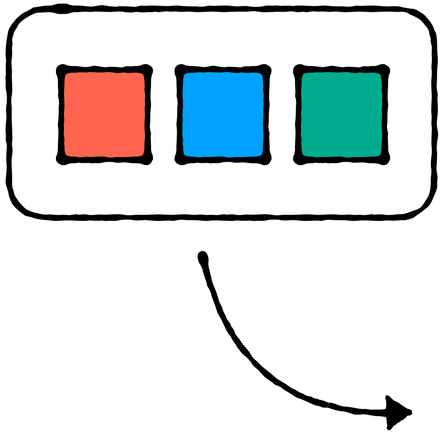
Tables



Storage

Buffer Management

Buffer Pool

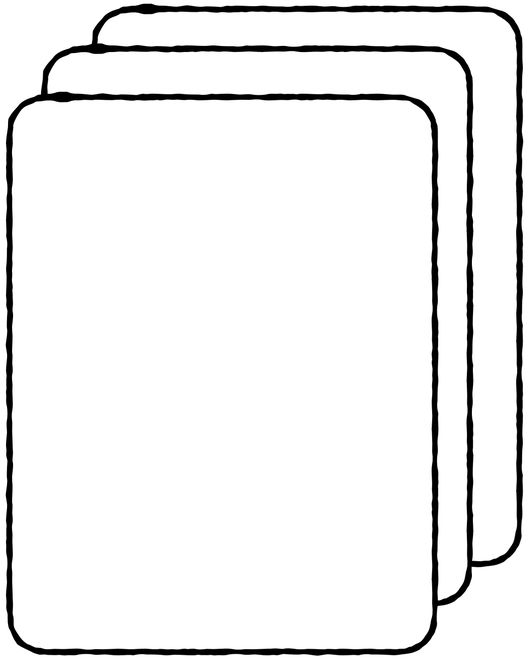


Which page to evict when
we run out of space?

Random, FIFO, LRU, MRU, Clock

Different trade-offs

Tables



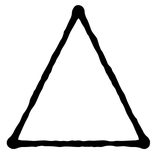
Storage

Storage

Tables

Buffer Management

Indexes



Sorting

Operators

Query Optimization

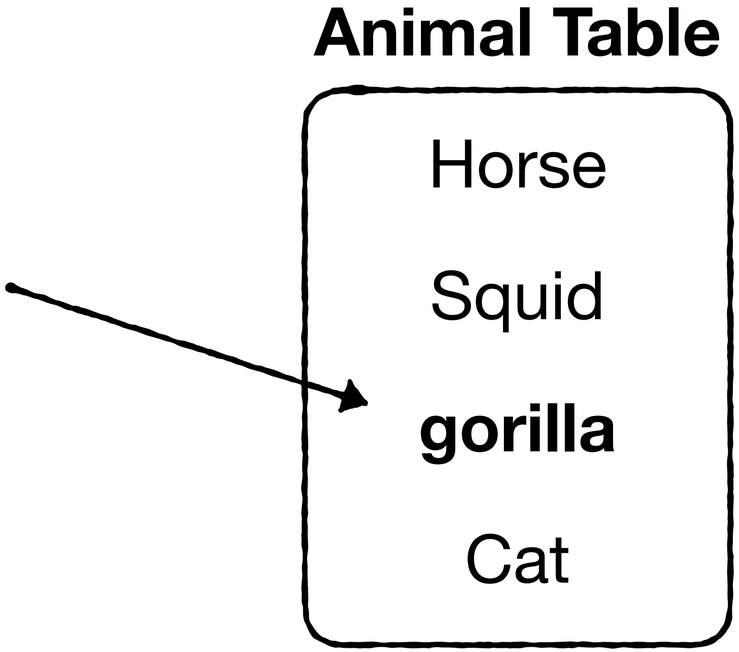
Transactions

Recovery

Indexes

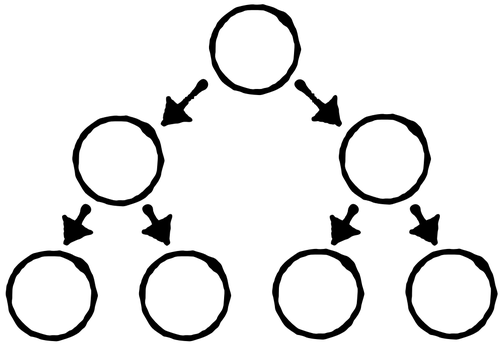
Select * from animals where name = “**gorilla**”

How to find quickly without a full scan?

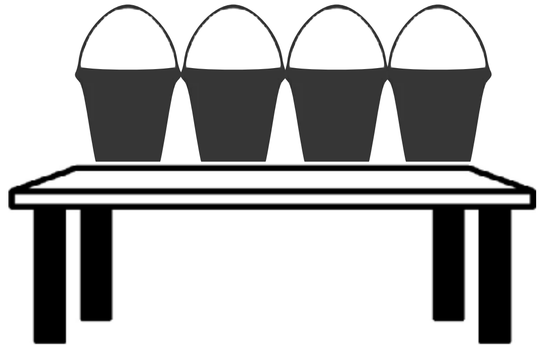


Indexes

You have learned about



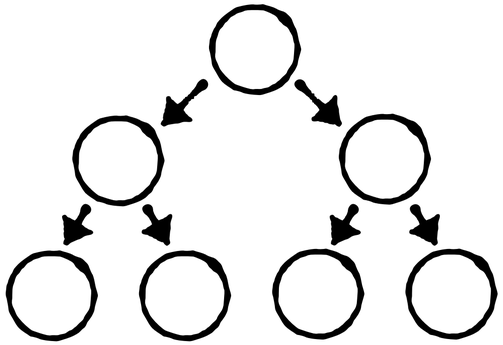
Binary trees



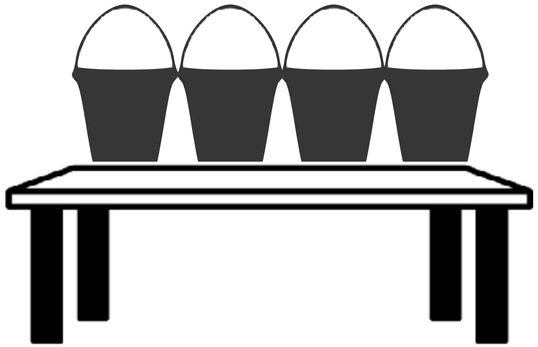
Hash tables

Indexes

You have learned about



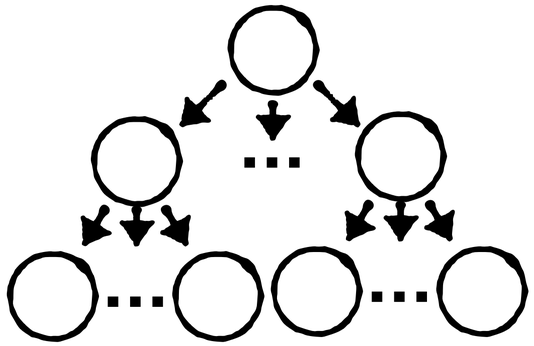
Binary trees



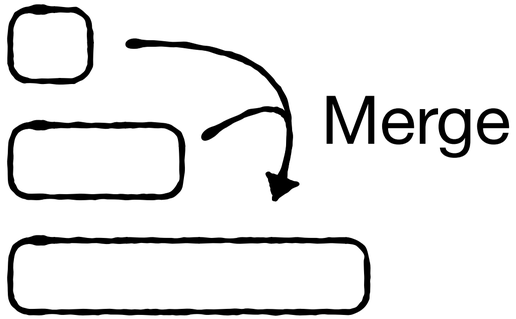
Hash tables

Not a good fit for disks or SSDs

Indexes

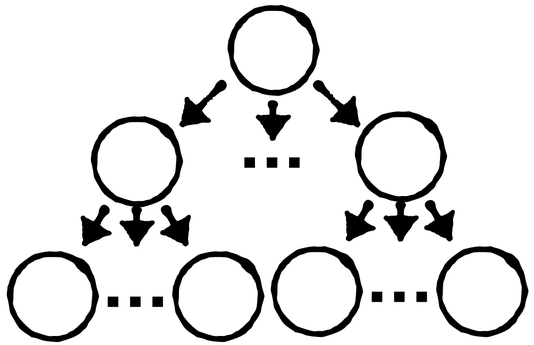


B-Trees

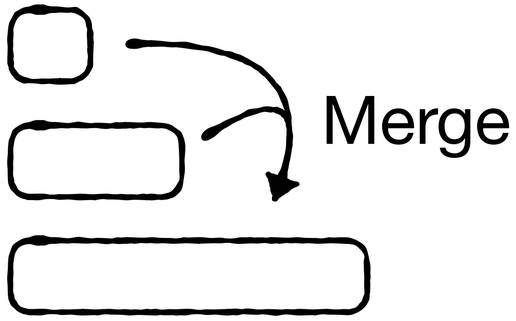


Log-Structured Merge-Trees

Indexes



B-Trees



Log-Structured
Merge-Trees

Cheaper queries

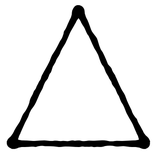
More memory
Cheaper writes

Storage

Tables

Buffer Management

Indexes



Sorting

Operators

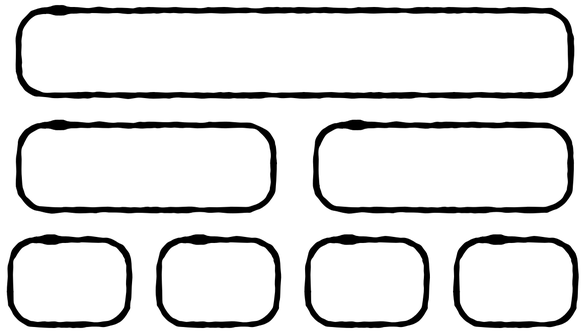
Query Optimization

Transactions

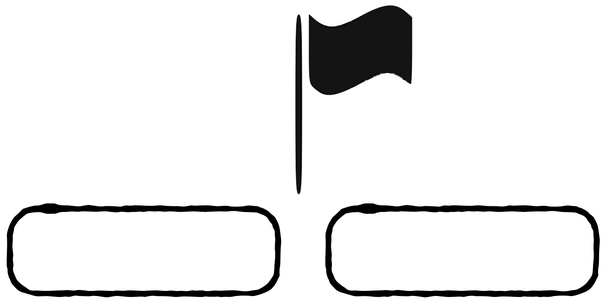
Recovery

Sorting

You have learned about



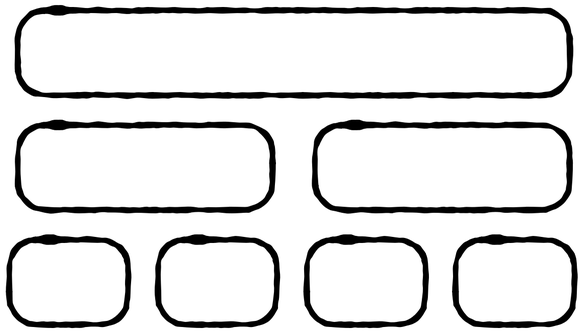
Merge-Sort



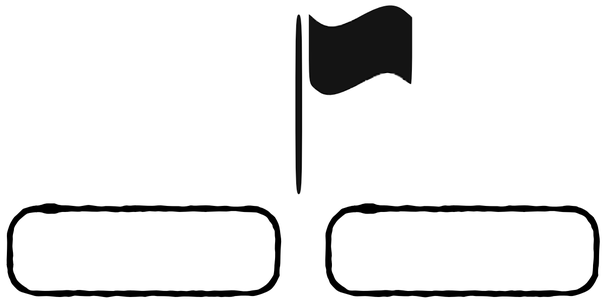
Quick-Sort

Sorting

You have learned about



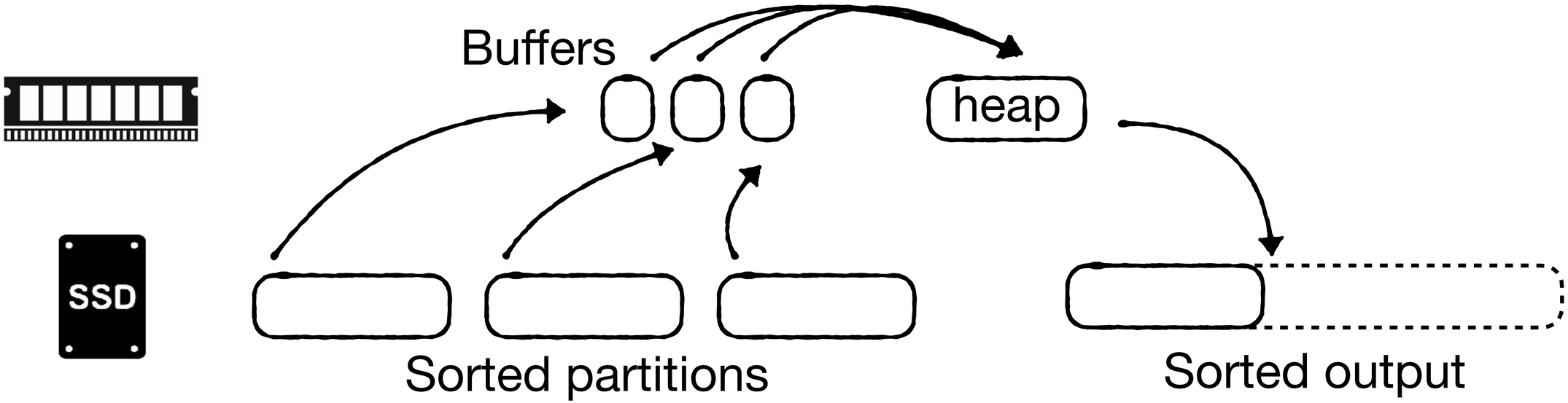
Merge-Sort



Quick-Sort

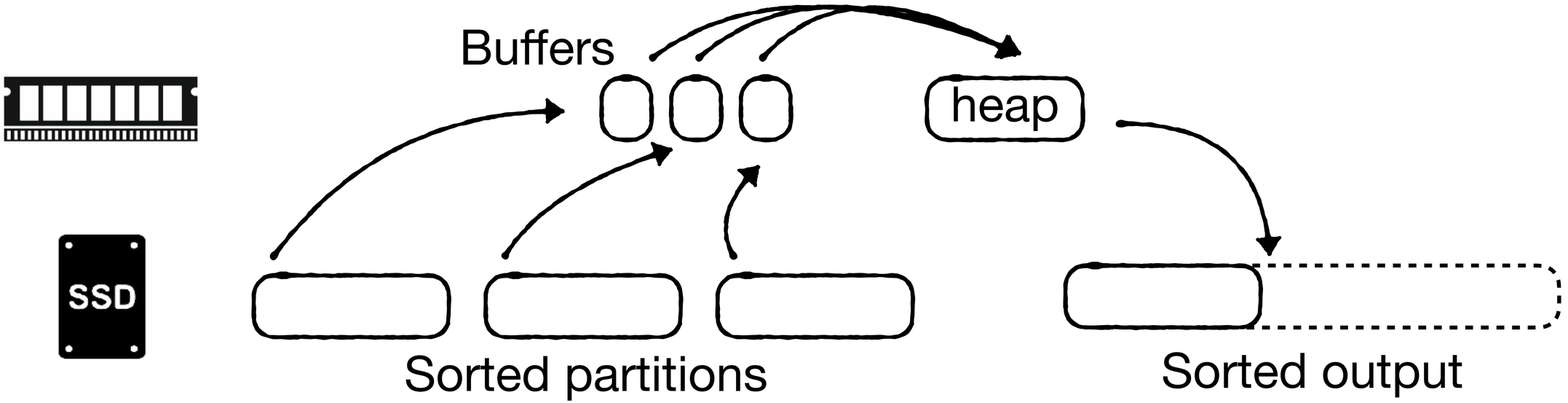
Also not a good fit for disks or SSDs

Sorting



Multi-way merge-sort

Sorting



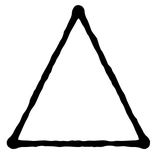
Sorts massive amounts of data in storage efficiently!

Storage

Tables

Buffer Management

Indexes



Sorting

Operators

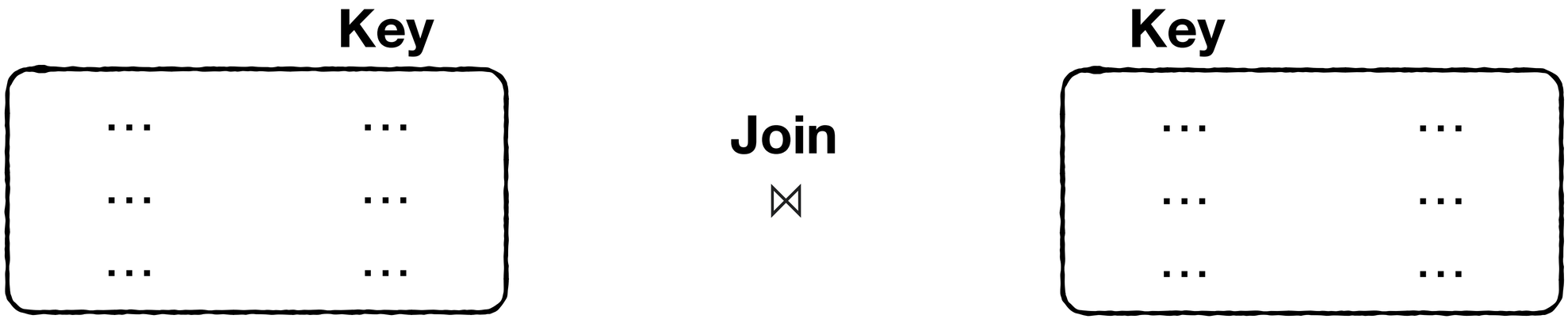
Query Optimization

Transactions

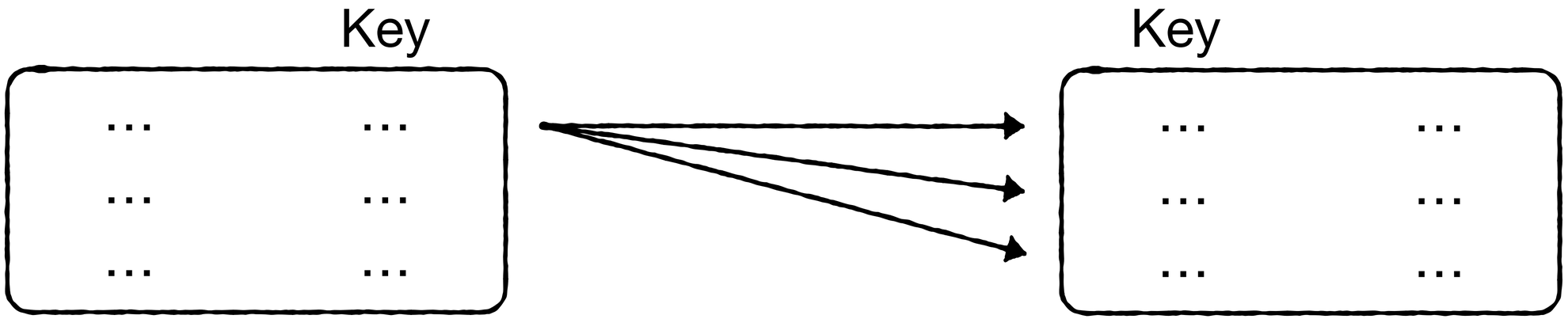
Recovery

Relational Operators: The algorithms used to physically answer queries

Relational Operators: The algorithms used to physically answer queries

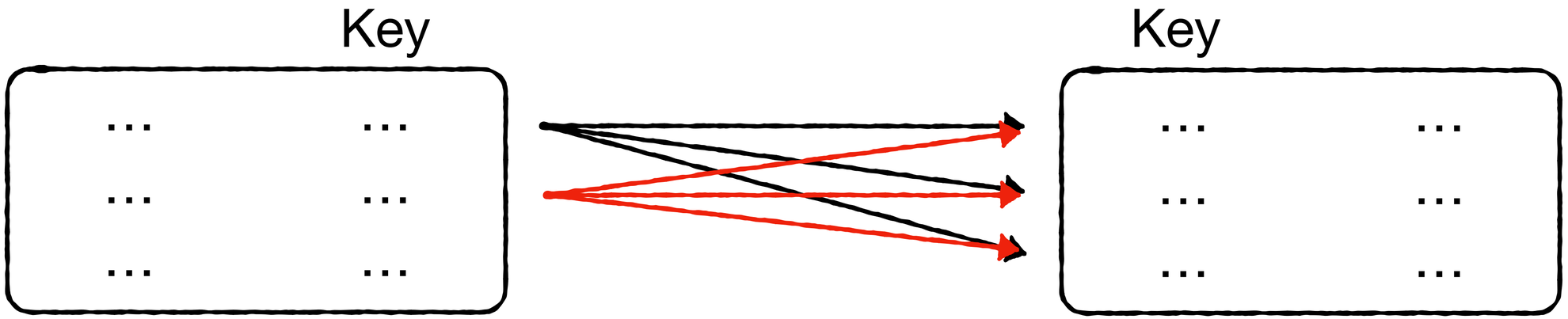


Relational Operators: The algorithms used to physically answer queries



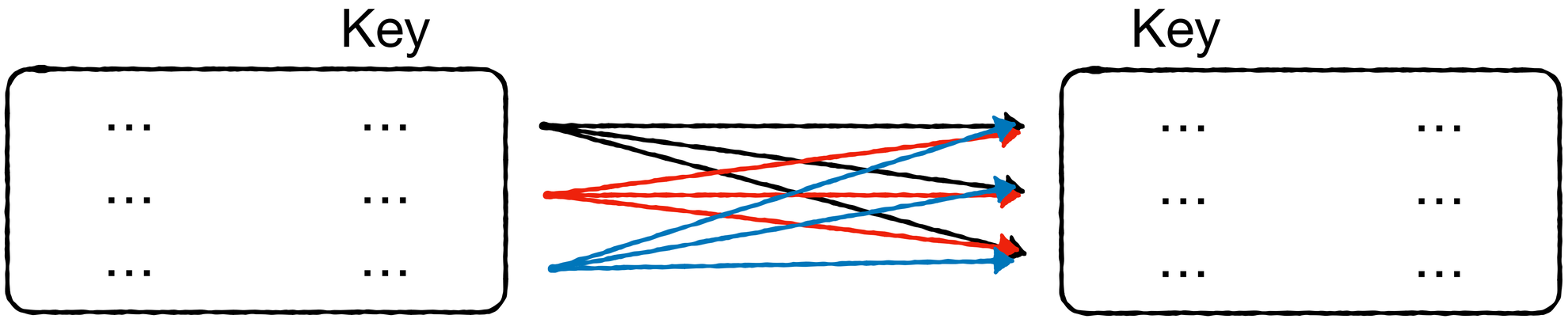
Nested join?

Relational Operators: The algorithms used to physically answer queries



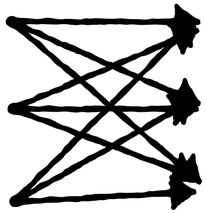
Nested join

Relational Operators: The algorithms used to physically answer queries

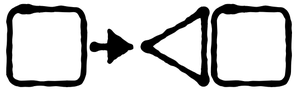


Nested join

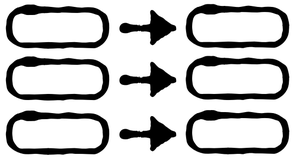
Relational Operators: The algorithms used to physically answer queries



Nested join



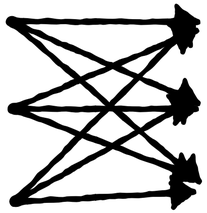
Index join



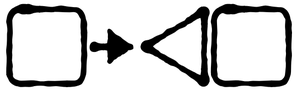
Grace Hash join

etc…

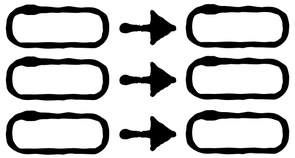
Relational Operators: The algorithms used to physically answer queries



Nested join



Index join



Grace Hash join

etc…

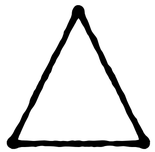
All have different properties and trade-offs :)

Storage

Tables

Buffer Management

Indexes



Sorting

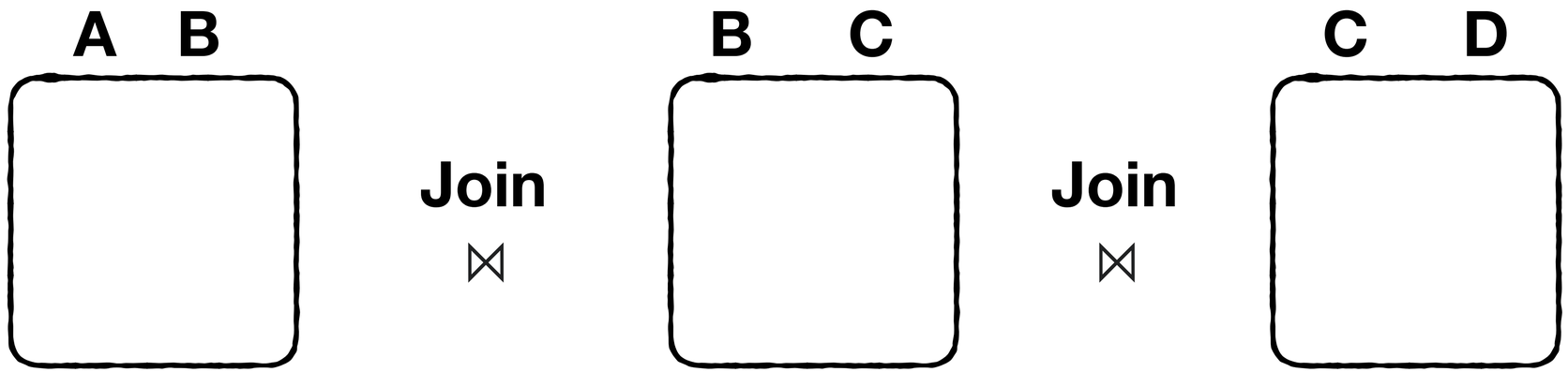
Operators

Query Optimization

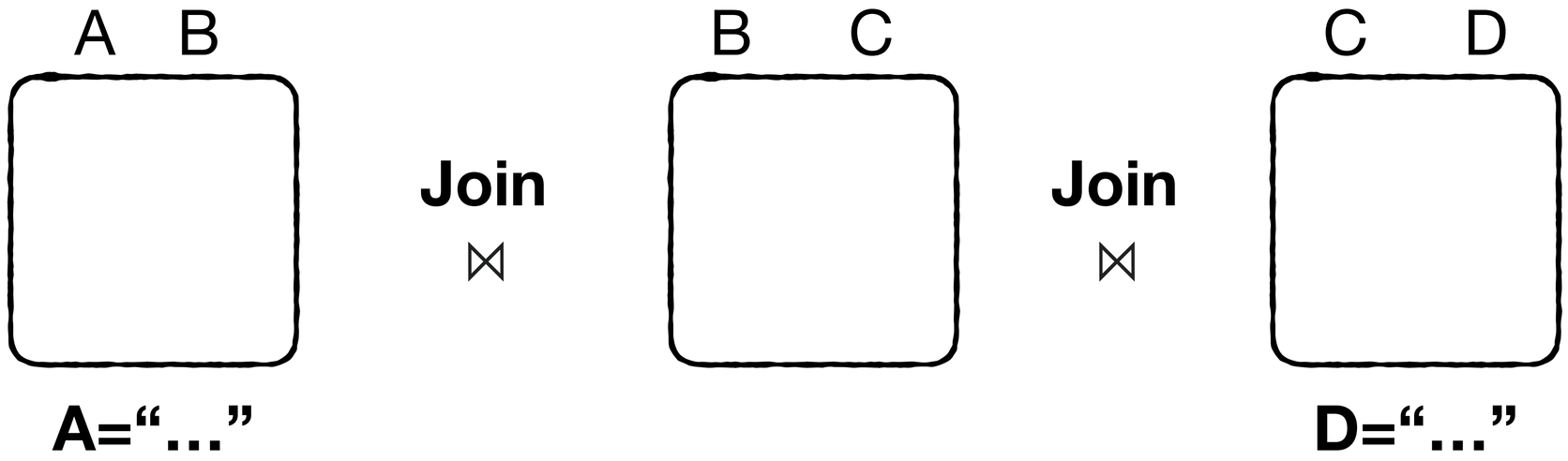
Transactions

Recovery

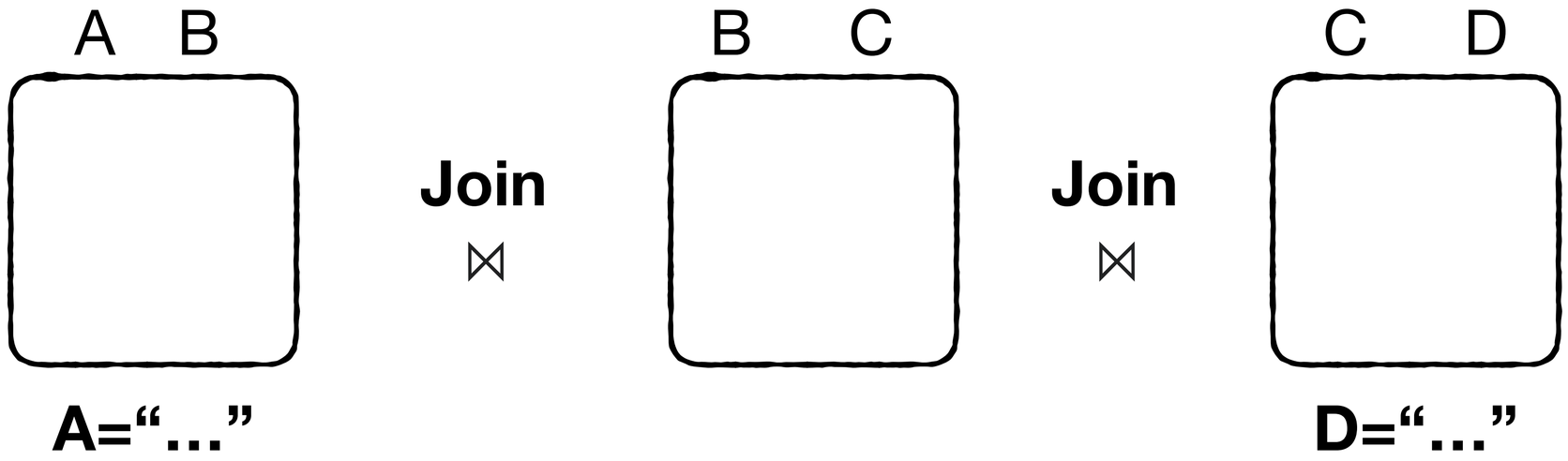
Query Optimization: constructing efficient query plans



Query Optimization: constructing efficient query plans

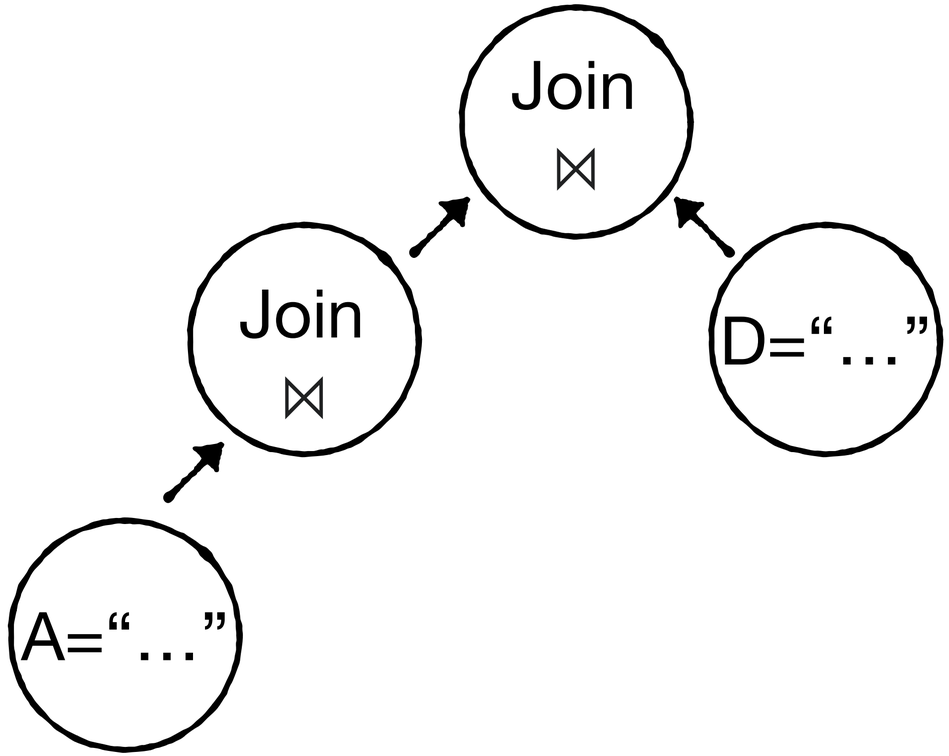


Query Optimization: constructing efficient query plans

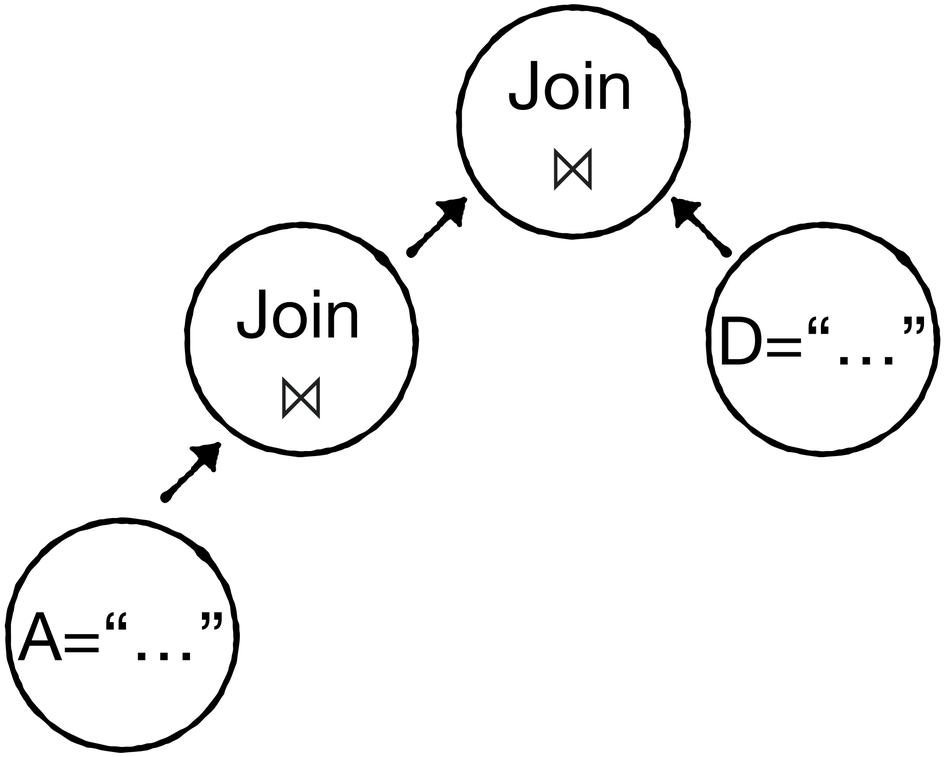


In which order to perform these operations?

Query Optimization: constructing efficient query plans



Query Optimization: constructing efficient query plans



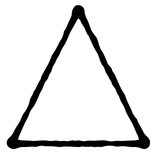
Using statistics & heuristics

Storage

Tables

Buffer Management

Indexes



Sorting

Operators

Query Optimization

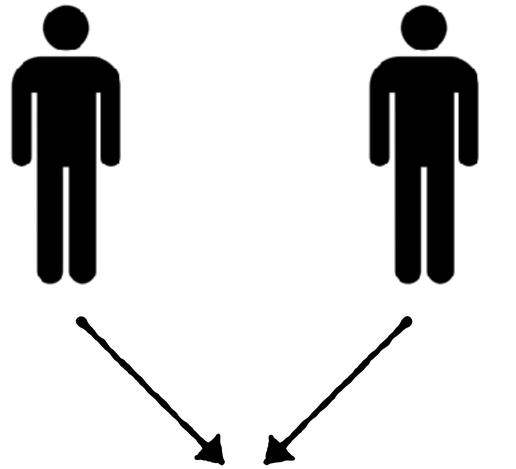
Transactions

Recovery

Transactions: ensuring complex operations run correctly, concurrently, and with fail-safety

Transactions: ensuring complex operations run correctly, concurrently, and with fail-safety

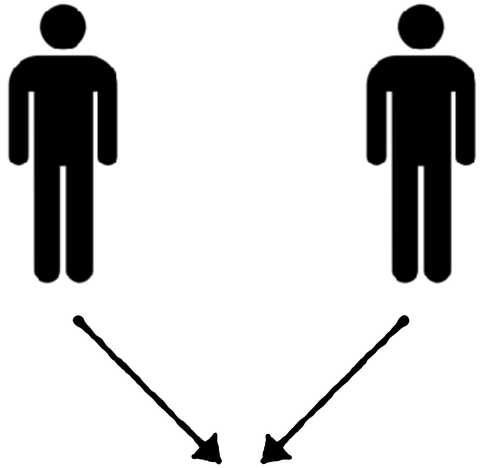
Two customers paying to an account at the same time can overwrite each other's changes



Account balance

Transactions: ensuring complex operations run correctly, concurrently, and with fail-safety

Two customers paying to an account at the same time can overwrite each other's changes

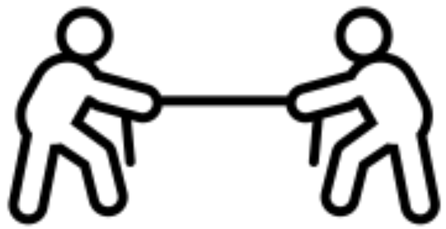


Account balance

How to maximize parallelism while maintaining correctness?

Transactions: ensuring complex operations run correctly, concurrently, and with fail-safety

parallelism



correctness

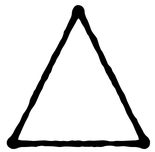
Trade & tune between these

Storage

Tables

Buffer Management

Indexes



Sorting

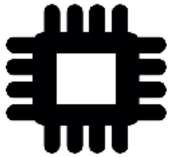
Operators

Query Optimization

Transactions

Recovery

Recovery



CPU caches

memory

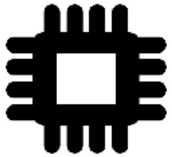
SSD

disk

volatile

Non-volatile

Recovery



CPU caches

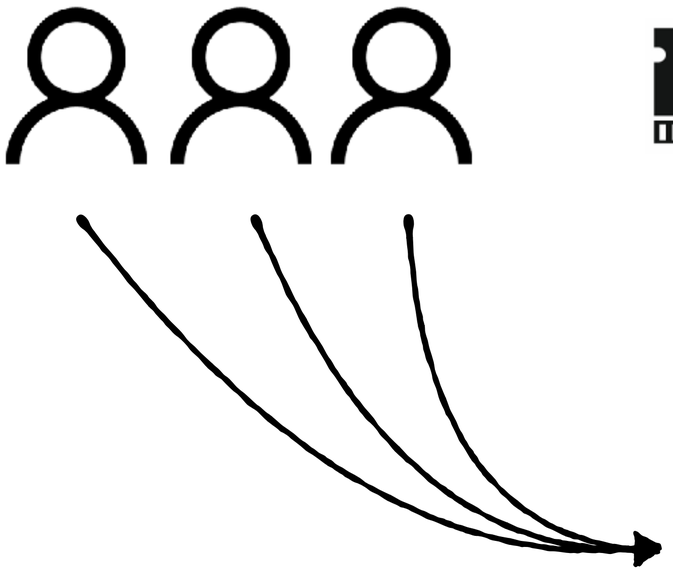
memory

SSD

disk

Data resides here for persistence

Recovery



memory

It is brought here to be updated

SSD

disk

Recovery

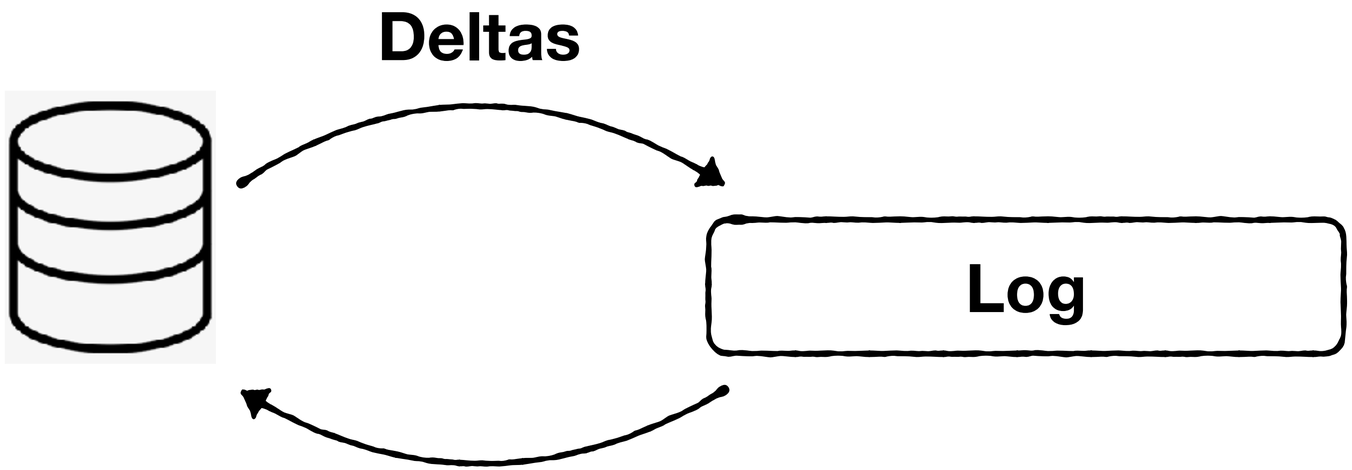
memory

SSD

disk

If power now fails, we lose everything

Recovery



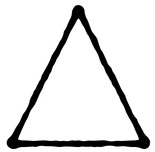
Efficient recovery

Storage

Tables

Buffer Management

Indexes



Sorting

Operators

Query Optimization

Transactions

Recovery

And now let’s get into storage