CS660: Intro to Database Systems

Database System Architectures

Instructor: Manos Athanassoulis

https://bu-disc.github.io/CS660/

Today

when you see this, I want you to speak up! [and you can always interrupt me]

logistics, goals, admin

database systems architectures

project info

Course Scope

A detailed look "under the hood" of a DBMS

why?

applications writers, data scientists

database researchers, db admins

they all *understand* the internals

there is a big need for **database systems experts**

data-intensive applications

big data workflows

Course Scope: Practical Side

query

build, design, & benchmark

understand

database systems!

More details when discussing the project!

Readings

“Cowbook”

by Ramakrishnan & Gehrke

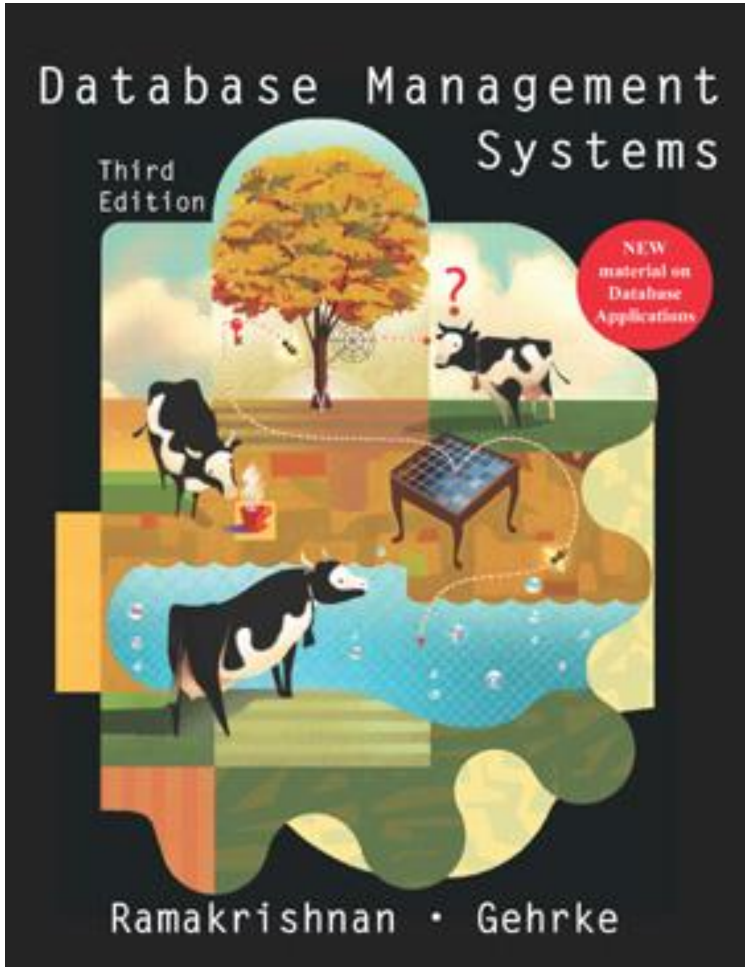
Additional Readings

Architecture of a Database System, by J. Hellerstein, M. Stonebraker and J. Hamilton

The Design and Implementation of Modern Column-store Database Systems, by D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden

Modern B-Tree Techniques, by Goetz Graefe, *Foundations and Trends in Databases, 2011*

+research papers


Database Management Systems
Third Edition
NEW material on Database Applications
Ramakrishnan • Gehrke

Guest Lectures

We plan will have a couple guest lectures

Make sure to attend!

Will be notified ahead of time.

STAY TUNED

Evaluation

Class Participation: 5%

In-class discussion

&

Collaborative Notes

2-3 students take notes (2 days after class anybody can augment it)
Shared Google doc: https://tinyurl.com/CS660-F24-Notes
[top part of website as well]

Enroll right after class!

Evaluation

Class Participation: 5%

Written Assignments: 10%

Graded on completion-basis

if you submit on time & >70% you get full credit

the goal of the assignments is to get familiar with exam-like questions

Throughout the semester

6 deadlines spread across the semester

[tentative topics and deadlines in the website]

Evaluation

Class Participation: 5%

Written Assignments: 10%

Programming Assignments: 40%

Assignments throughout semester

[more details later today]

Evaluation

Class Participation: 5%

Written Assignments: 10%

Programming Assignments: 40%

Midterm: 20%

Final: 25%

(more details soon)

Evaluation

Class Participation: 5%

Written Assignments: 10%

Programming Assignments: 40%

Midterm: 20%

Final: 25%

SQL Hands-on Bonus: 5%

Office Hours

OH are in-person

(online OH can be arranged when needed)

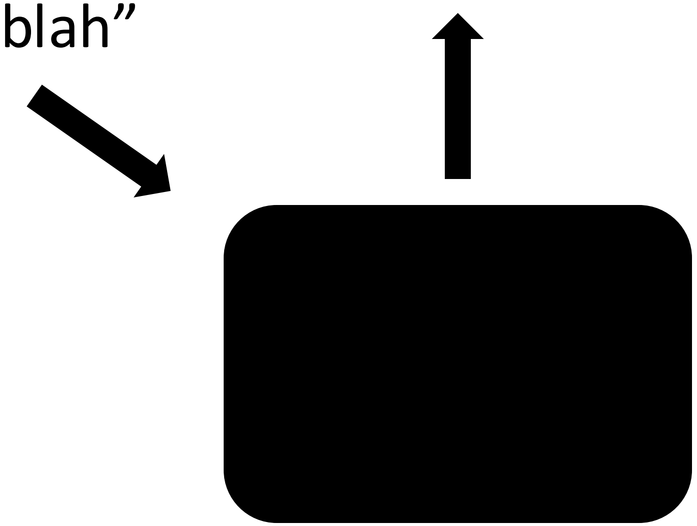
Manos

Tu @ 10am / Th @ 2pm (after class) in CCDS928

TAs

announced in Piazza

Database Systems



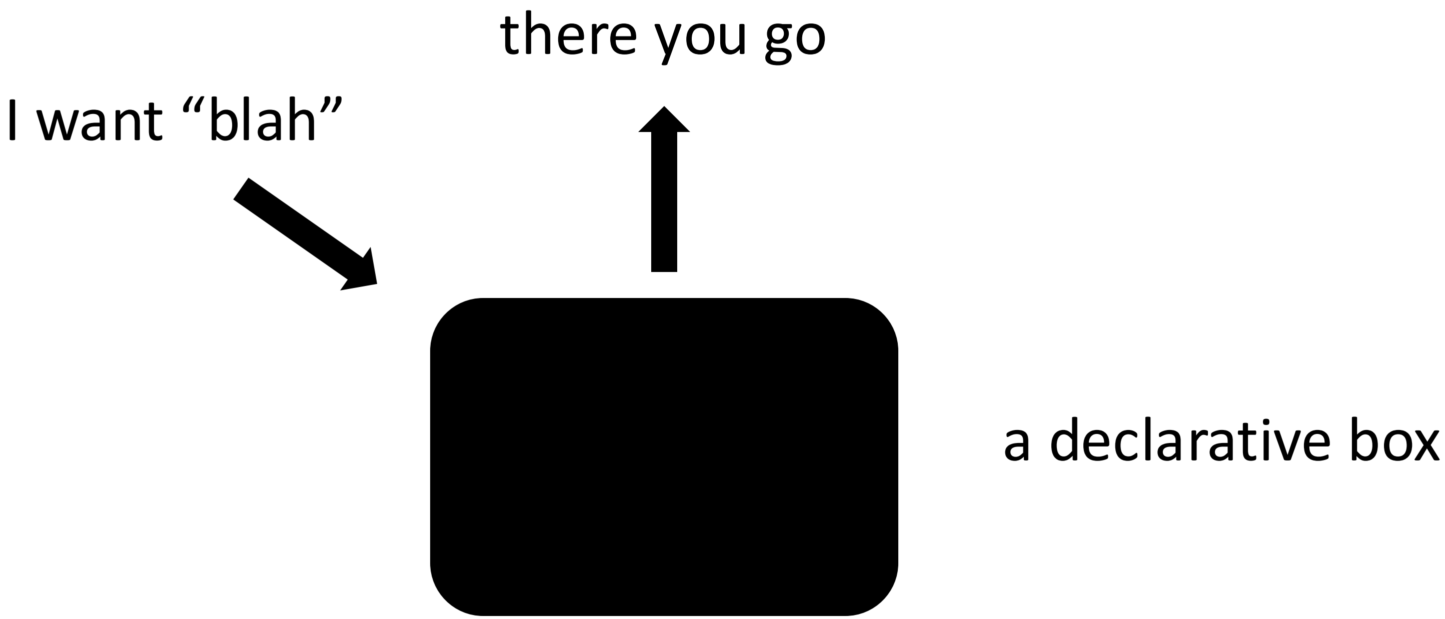
there you go

I want “blah”

a declarative box

why having a declarative box is useful?

Database Systems



there you go

I want “blah”

a declarative box

application and **backend** development are independent collection of algorithms & data structures

multiple ways to do the same thing

optimization: dynamically decide which to use

how?

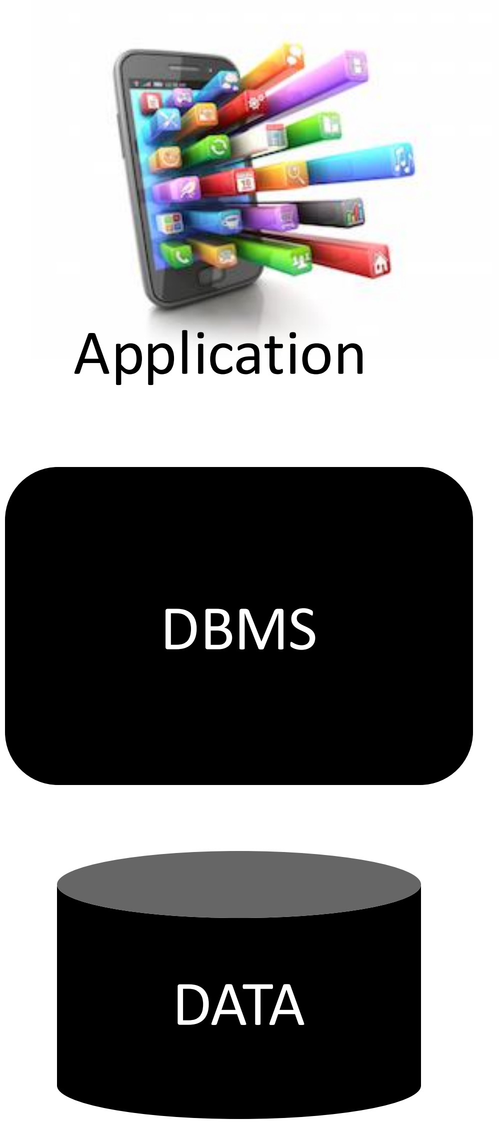
collection of algorithms & data structures

multiple ways to do the same thing

optimization: dynamically decide which to use

how? understand & model alternatives

data management goals



Application

DBMS

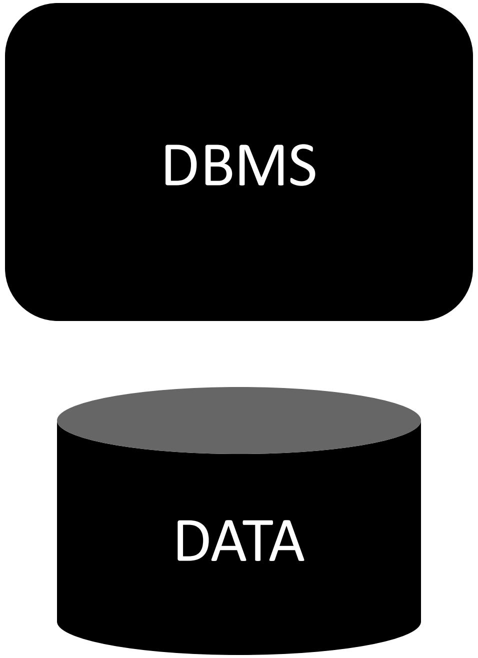
DATA

data management goals

Application

monetary cost

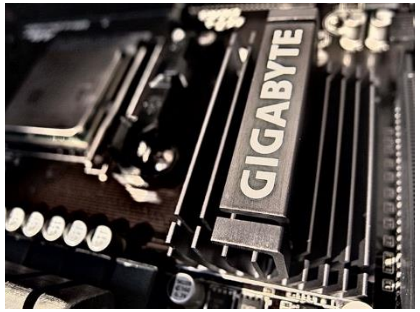
performance



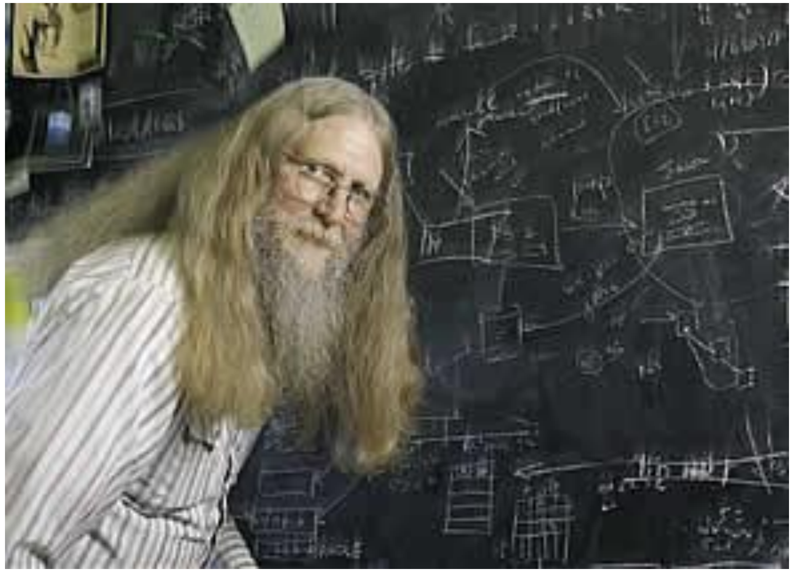
DBMS

DATA

energy



hardware

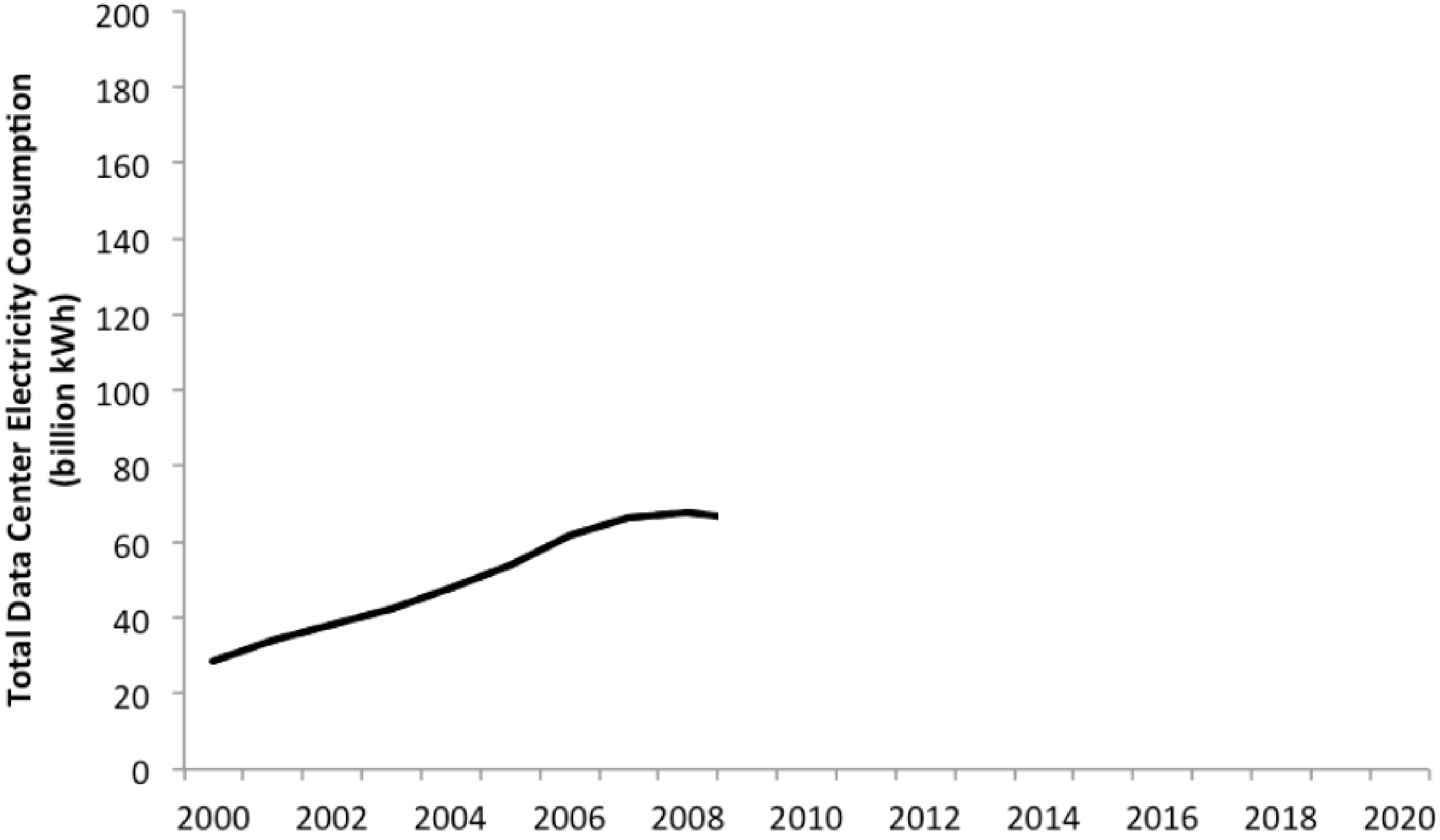


“three things are important in the database world: ***performance, performance,*** *and* ***performance****”*

Bruce Lindsay, IBM Research

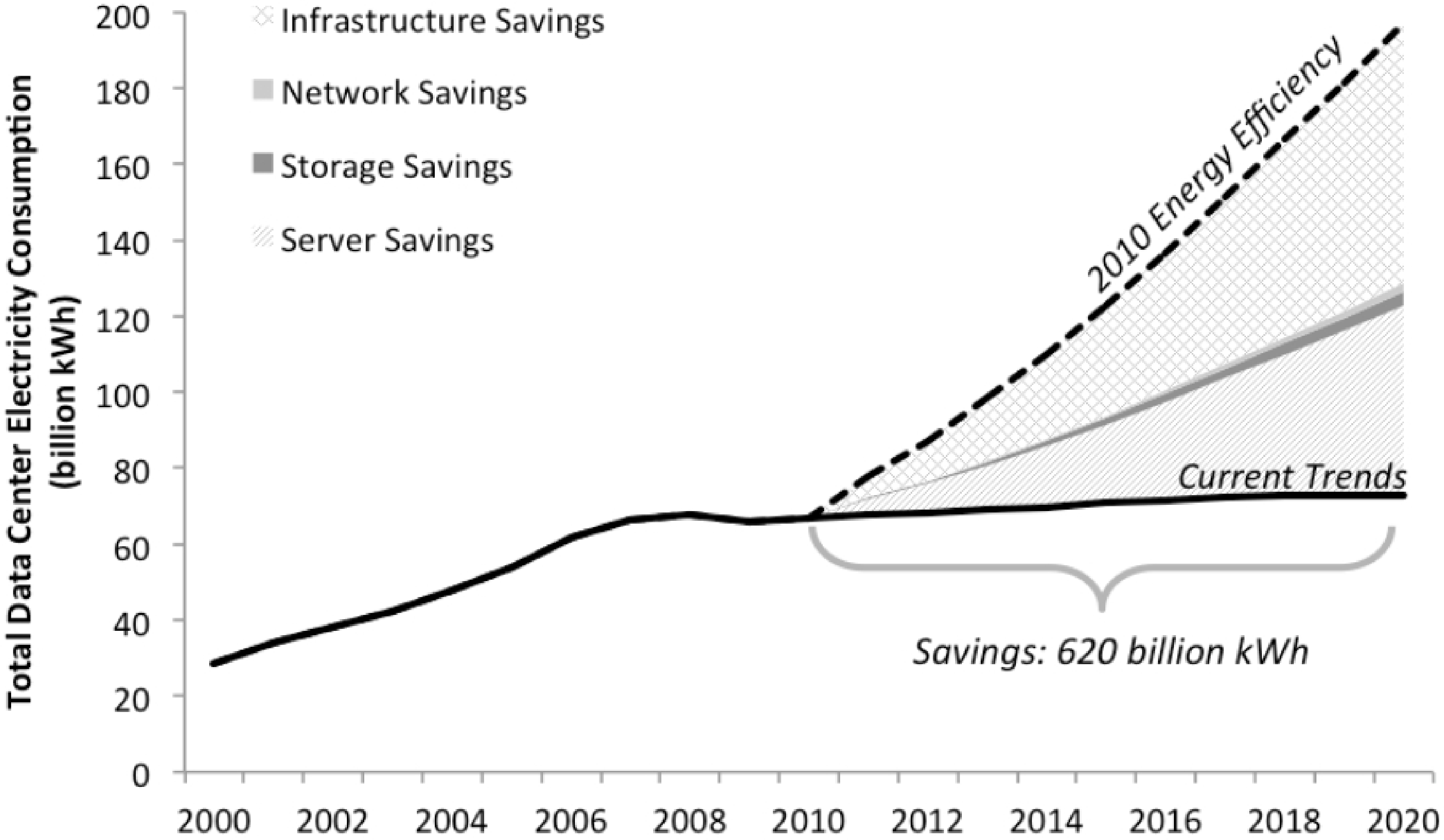
ACM SIGMOD Edgar F. Codd Innovations award 2012

but



datacenterknowledge.com, 2016

but



datacenterknowledge.com, 2016

but

new hardware in the last 20 years

multi-core processors

multi-level cache memories

flash drives

SIMD instructions

…

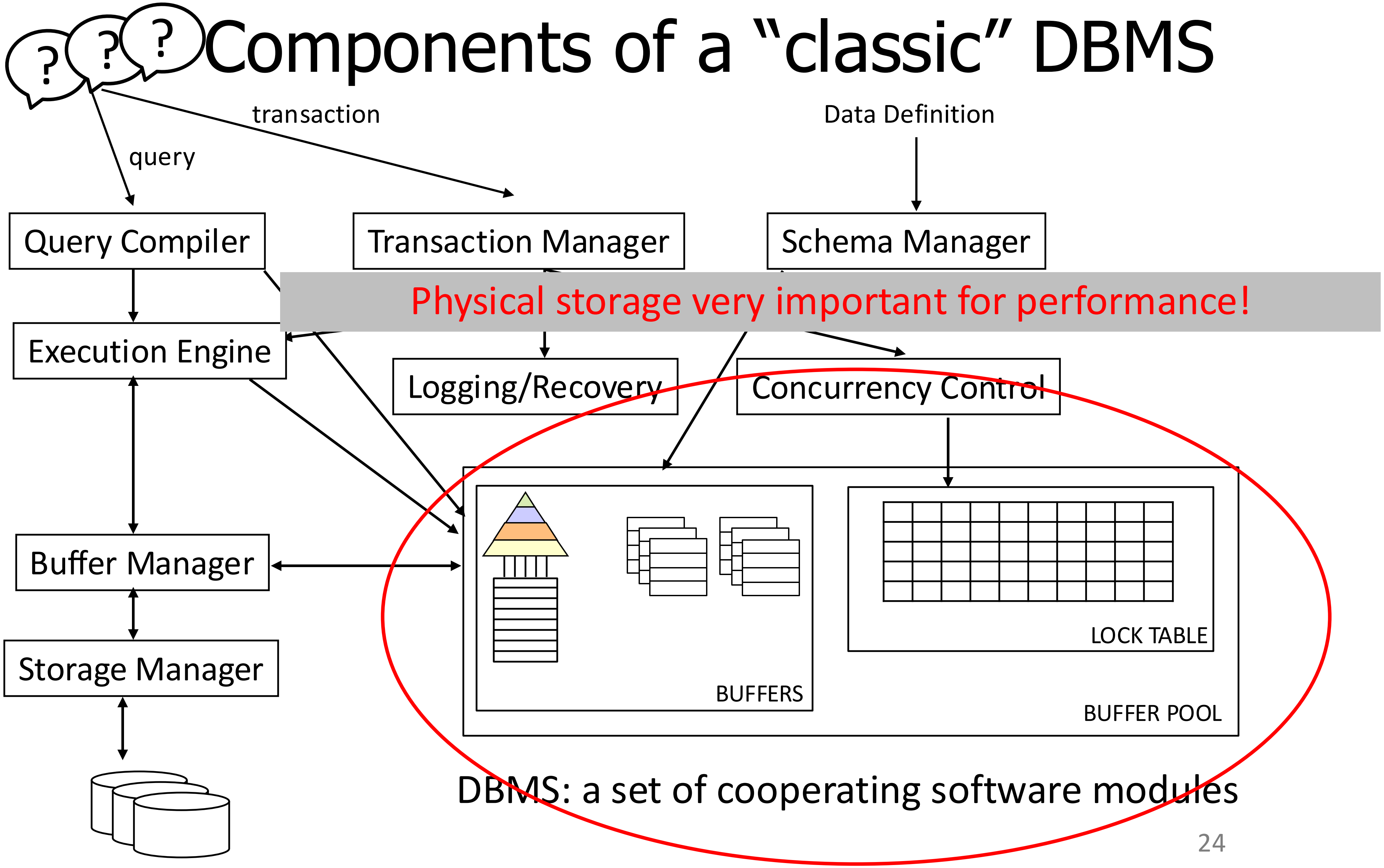
CS660

What is inside?

How it works?

performance on
a declarative box

Components of a “classic” DBMS



Physical storage very important for performance!

DBMS: a set of cooperating software modules

Some questions for today

how can we physically store our (relational) data?

how to efficiently access the data?

does that affect the way we ***ask*** queries?

does that affect the way we ***evaluate*** queries?

does that affect the way we apply ***updates***?

how to physically store data?

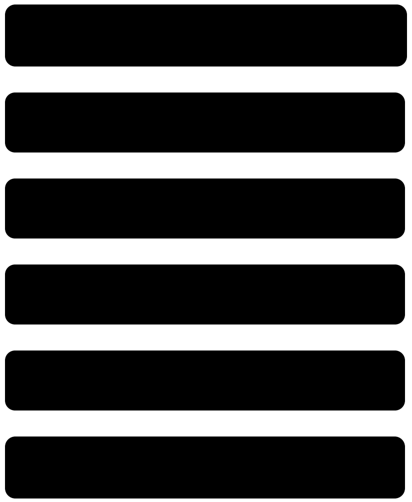
what is a relation?

a table with rows & columns!

how to physically store it?

how to physically store data?

one row at a time



how to efficiently access data?

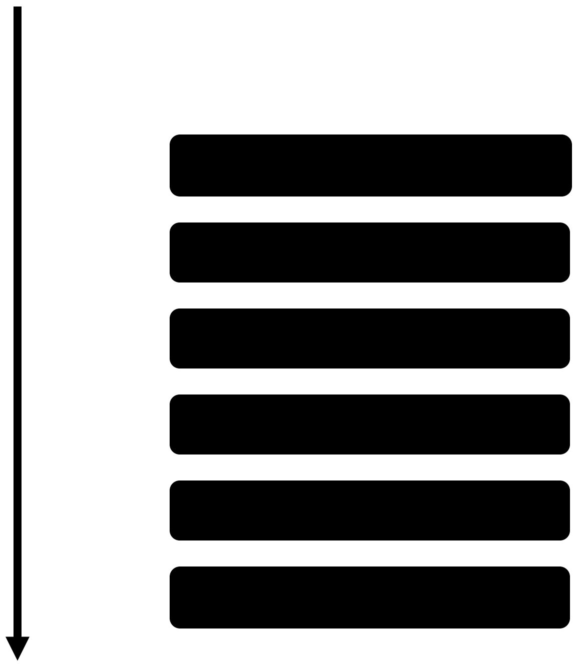
how to retrieve rows:

if I am interested in the average GPA of all students?

if I am interested in the GPA of student A?

how to efficiently access data?

Scan the whole table



if I am interested in most of the data

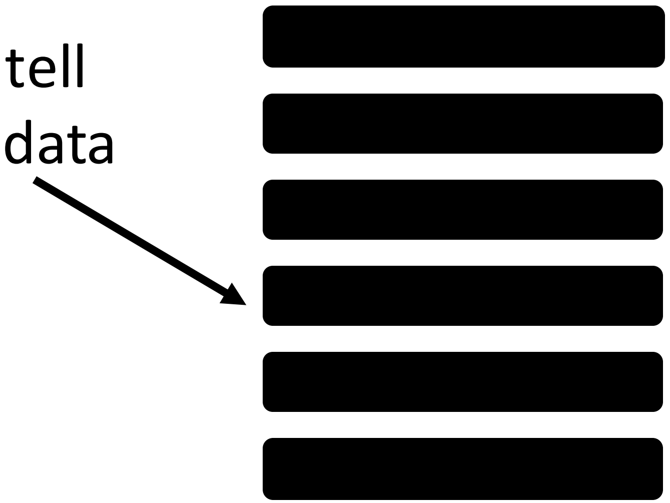
how to efficiently access data?

how to retrieve rows:

if I am interested in the average GPA of all students?

if I am interested in the GPA of student A?

how to efficiently access data?



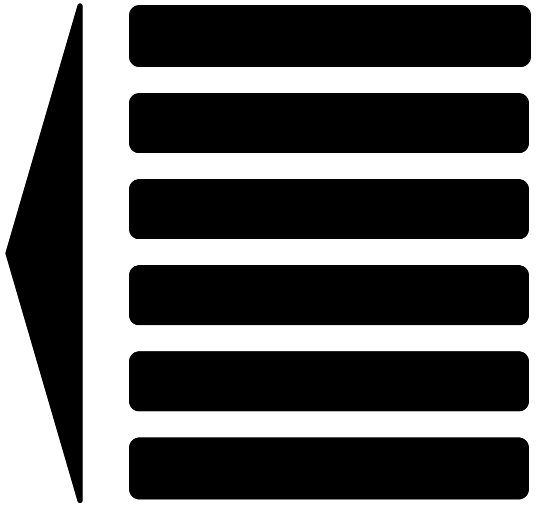
Ask an *oracle* to tell me where is my data

if I am interested in a single row

how to efficiently access data?

what is an *oracle* or *index*?

a data structure that given a value (e.g., student id)
returns location (e.g., row id or a pointer)
with less than O(n) cost ideally O(1)!



e.g., B Tree, bitmap, hash index

examples?

how to efficiently access data?

Scan vs. Index

How to choose?

Model!

What are the parameters?

data size

index traversal cost

access cost (random vs. sequential)

result set size ("selectivity")

Query Optimization!

how to efficiently access data?

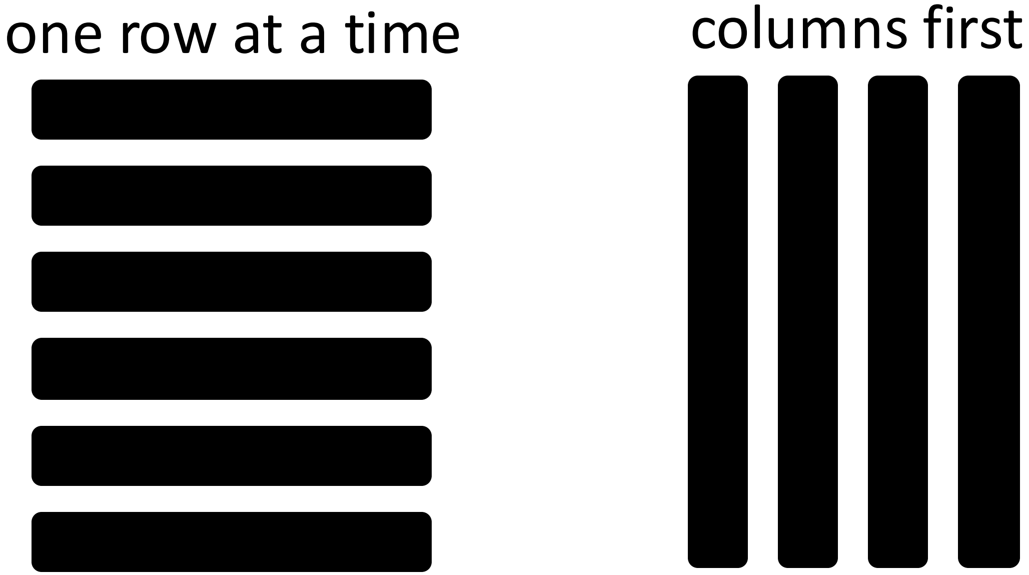
Scan vs. Index

Scan: many rows

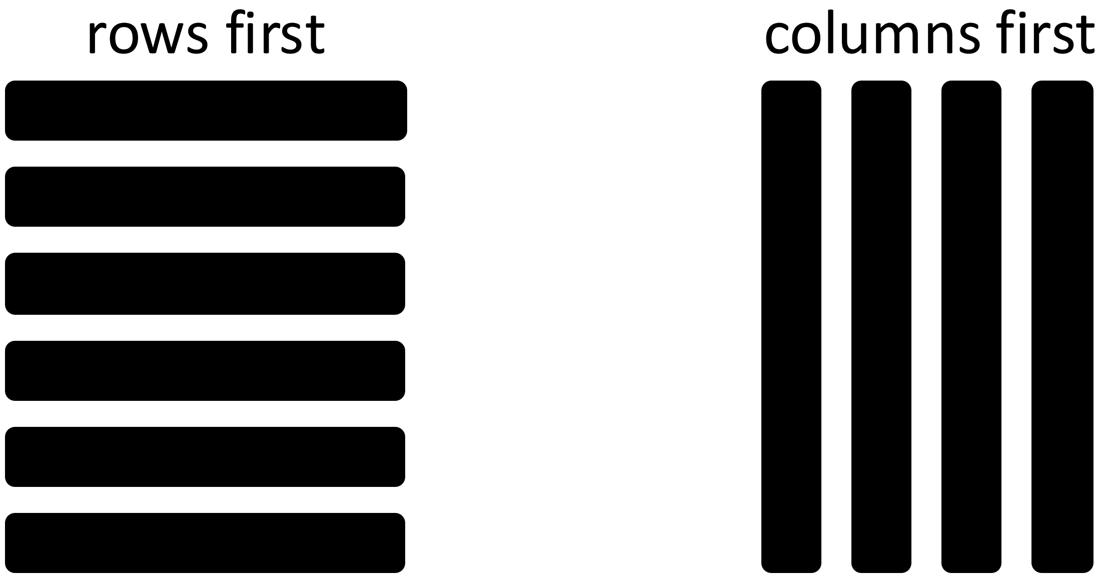
Index: few rows

how to physically store data?

is there another way?



how to efficiently access data?



if I want to access all the information of a single student?

if I want to find the name of the younger student?

if I want to calculate the average GPA?

if I want the average GPA of all students with CS Major?

how to efficiently access data?

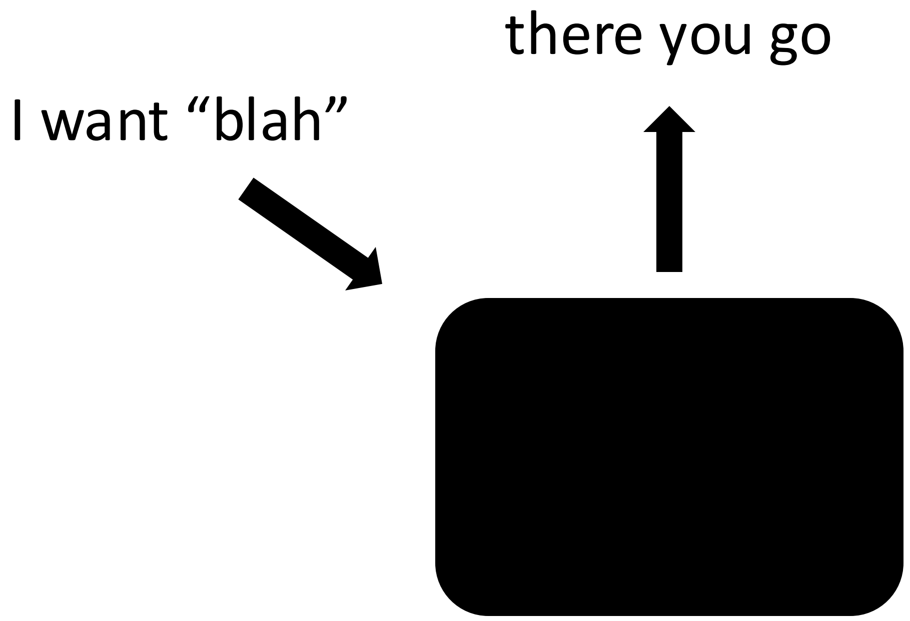
Rows vs. Columns

Rows: many attributes + few rows

Columns: few attributes + lots of rows

does that affect the way we ***ask*** queries?

No!



a declarative box

does that affect the way we ***evaluate*** queries?

Query Engine *is* different

row-oriented systems (”row-stores”)
move around rows

column-oriented systems (”column-stores”)
move around columns

does that affect the way we ***evaluate*** queries?

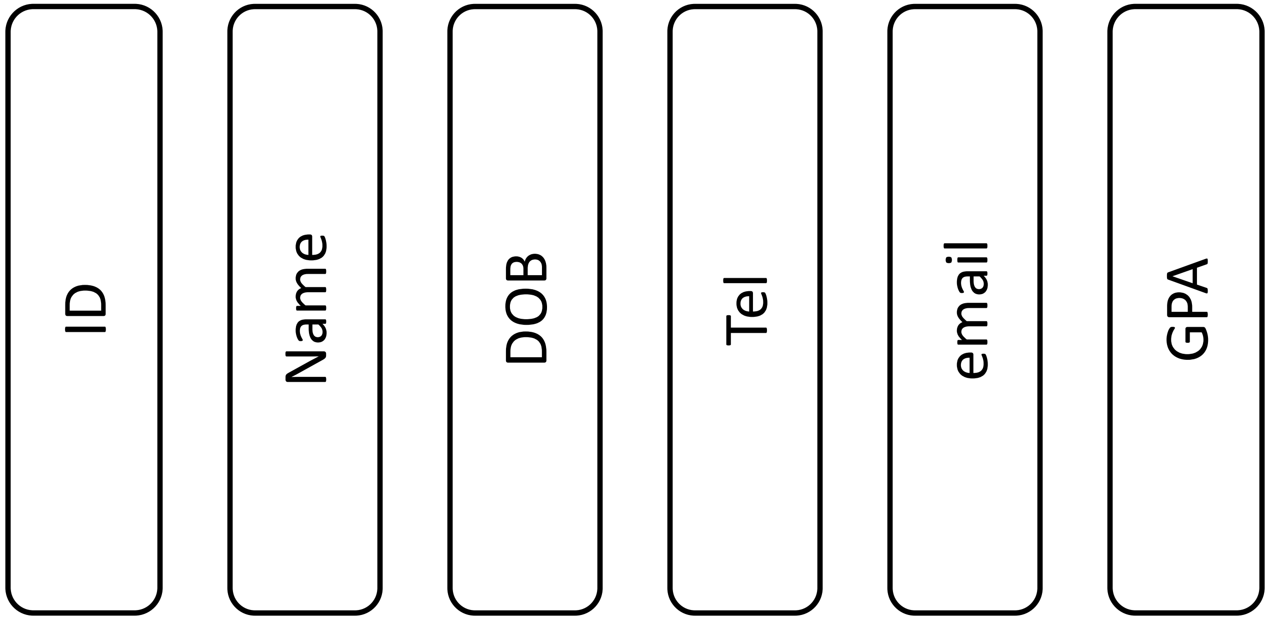
ID | Name | DOB | Tel | email | GPA

easy mapping from SQL to evaluation strategy

few basic operators: select, project, join, aggregate

simple logic for “query plan”

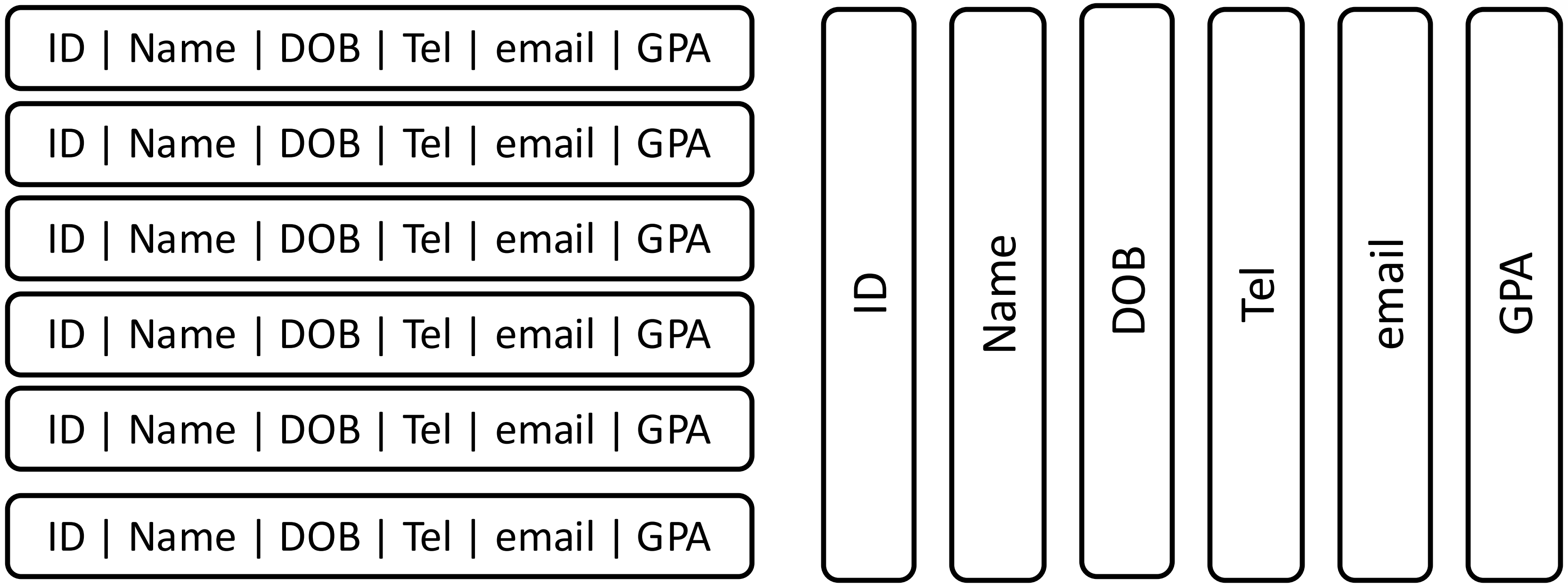
does that affect the way we ***evaluate*** queries?



simpler basic operators

complicated query logic (more operators to connect)

does that affect the way we apply ***updates***?



ID | Name | DOB | Tel | email | GPA

ID | Name | DOB | Tel | email | GPA

ID | Name | DOB | Tel | email | GPA

ID | Name | DOB | Tel | email | GPA

ID | Name | DOB | Tel | email | GPA

ID | Name | DOB | Tel | email | GPA

ID

Name

DOB

Tel

email

GPA

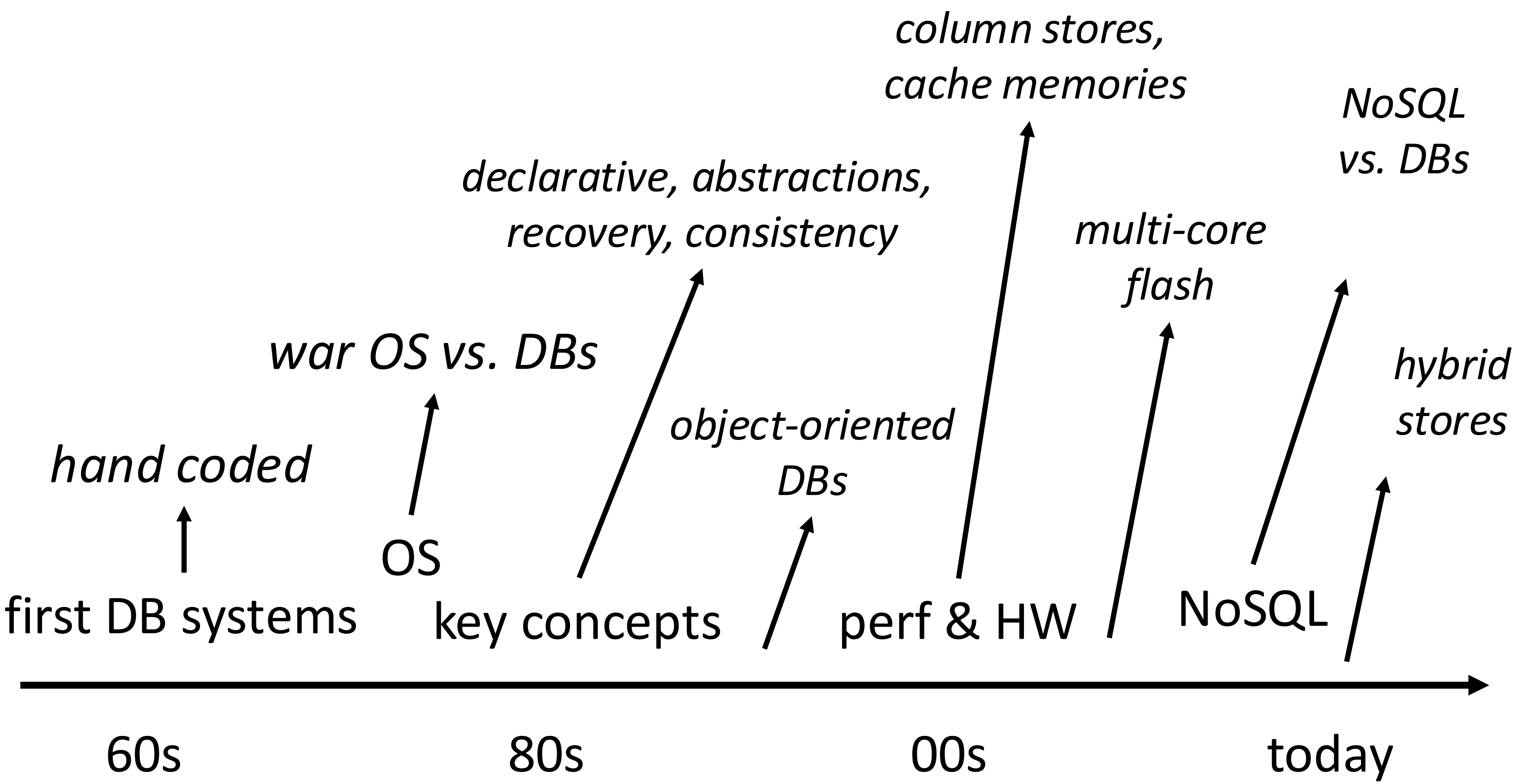
how to insert a new row?

how to delete a row?

how to change the GPA of a student?

how to update the email format of all students?

DBMS timeline



Row-Stores vs. Column-Stores

physical data layout

simple query plan vs. simple operators

“transactions” vs. ”analytics”

Other Architectures?

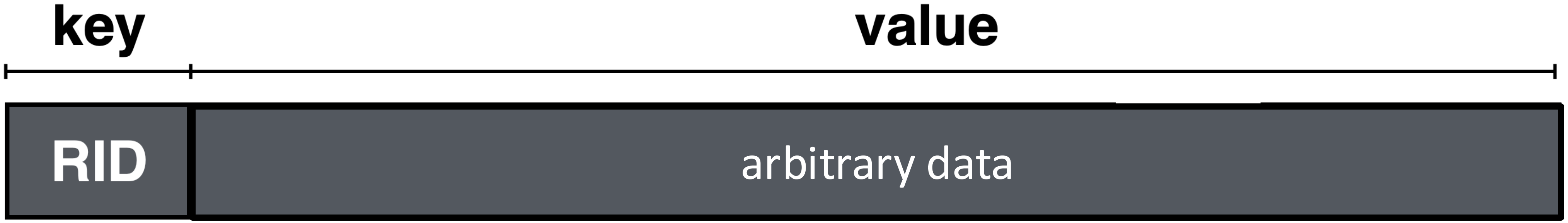
Key-Value Stores (NoSQL)

no transactions

data model: **keys** & **values**

row: a key and an *arbitrarily complex* value

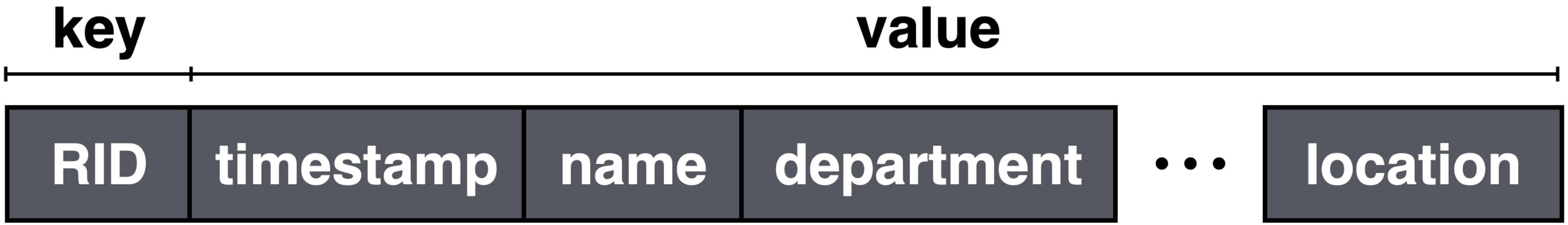
Key-Value Pair



semi-structured data

document data

Key-Value Pair



key

value

RID | timestamp | name | department | • • • | location

semi-structured data

document data

relational data

Other Architectures?

Key-Value Stores (NoSQL)

no transactions

data model: **keys** & **values**

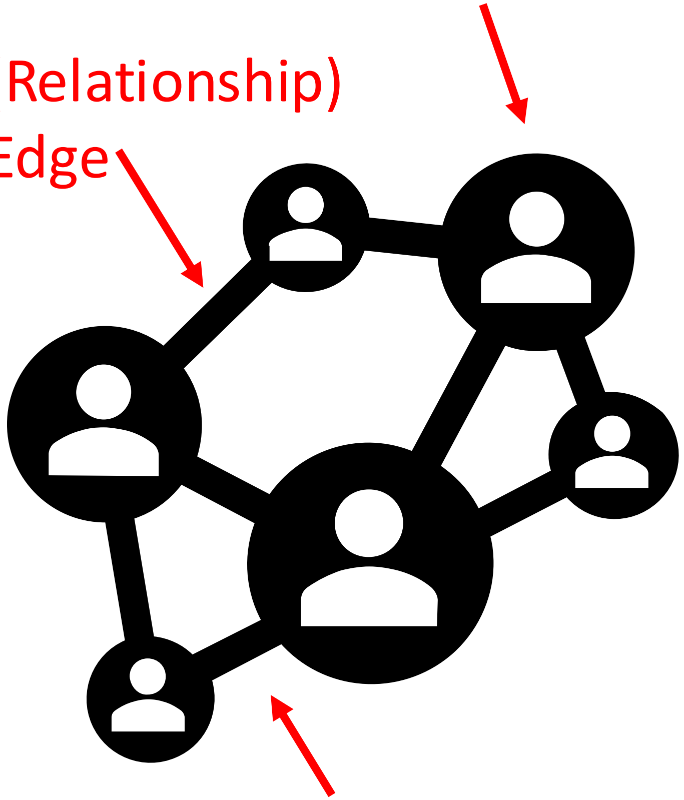
row: a key and an *arbitrarily complex* value

Graph Stores

natural representation of graph links

data model: **nodes** & **relationships**

also maybe: **weights, labels, properties**



Node

(Relationship)
Edge

Edges can have weights,
labels, or other properties

Programming Assignment (SimpleDB)

A basic DBMS developed by Sam Madden (MIT) for educational purposes

It has a SQL front-end

You will be implementing functionality in
(1) Bufferpool
(2) Heapfiles / Catalog / Tuple descriptor
(3) B-Tree Indexes
(4) Query Processing
(5) Query Optimization

project in groups of 2

Piazza

Announcements & Discussions in Piazza

https://piazza.com/bu/fall2024/cs660

Remember & Next Time

database systems: performance (but energy, HW)

physical storage (row-oriented vs. col-oriented)
affects query engine/big design space

Main Project: build a database system
More programming assignments later on

Next: SQL