

CS460: Intro to Database Systems

# *Database System Architectures*

Instructor: Manos Athanassoulis

<https://bu-disc.github.io/CS460/>

# Today



logistics, goals, admin

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

database systems architectures

project details

# 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 huge need for database experts  
data-intensive applications  
big data workflows

# Course Scope: Practical Side

use



benchmark



understand



database systems!

More details when discussing the project!

# Readings

## “Cowbook”

by Ramakrishnan & Gehrke

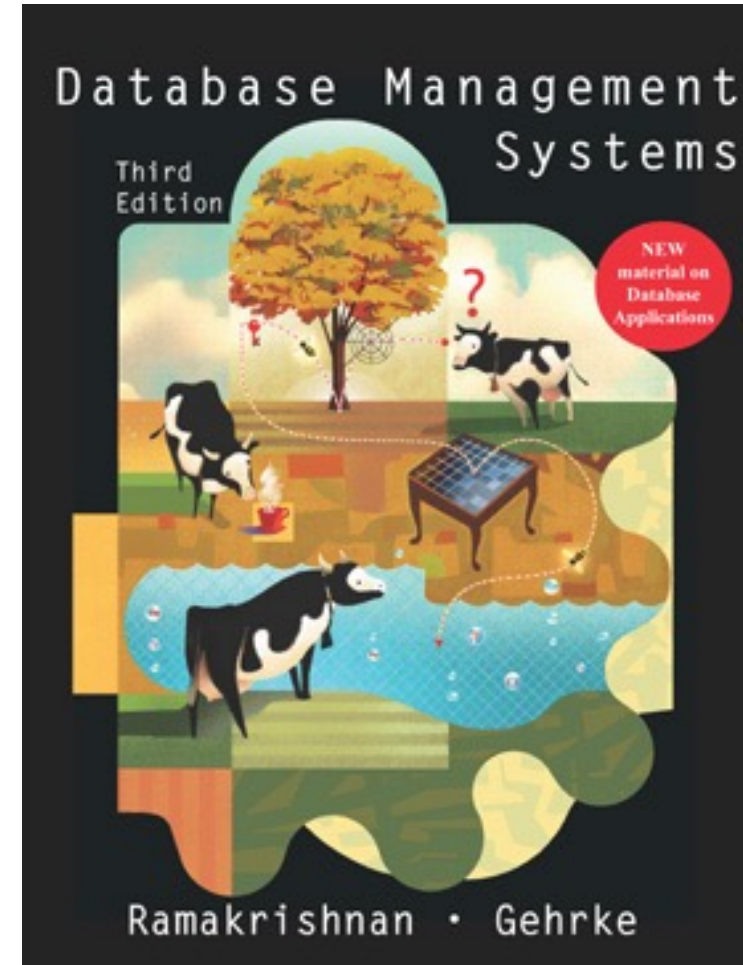
## Additional Readings

[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



# 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/CS460-F21-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**

7 deadlines spread across the semester

[topics and deadline soon in the website]

# Evaluation

Class Participation: 5%

Written Assignments: 10%

Programming Assignments: 30%

**Three assignments throughout semester**

[more details later today]



# Evaluation

Class Participation: 5%

Written Assignments: 10%

Programming Assignments: 30%

Midterm: 20%

Final: 35%

(more details soon)

# Evaluation

Class Participation: 5%

Written Assignments: 10%

Programming Assignments: 30%

Midterm: 20%

Final: 35%

**Hands-on Bonus: 5% (SQL 3% & Key-value quering 2%)**

# Office Hours

**OH are in-person**

(online OH can be arranged when needed)

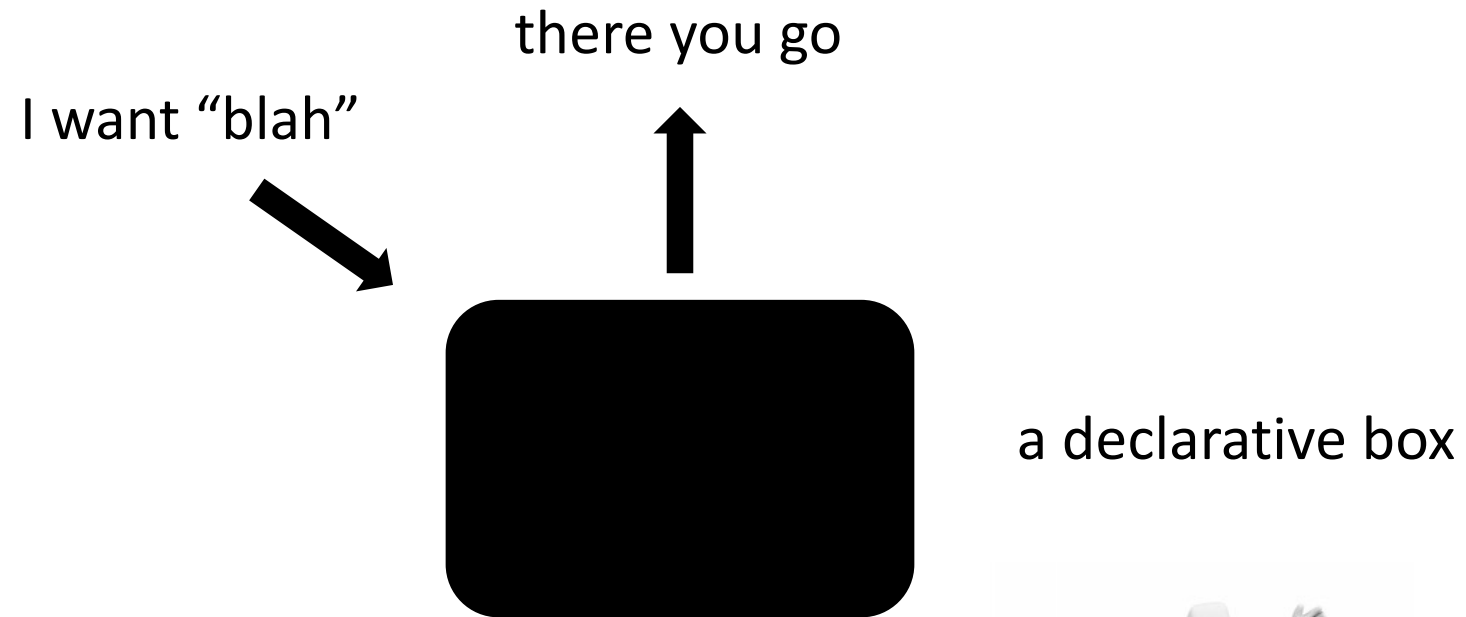
**Manos**

T/Th @ 3:30pm (after class)

**TAs**

announced in Piazza

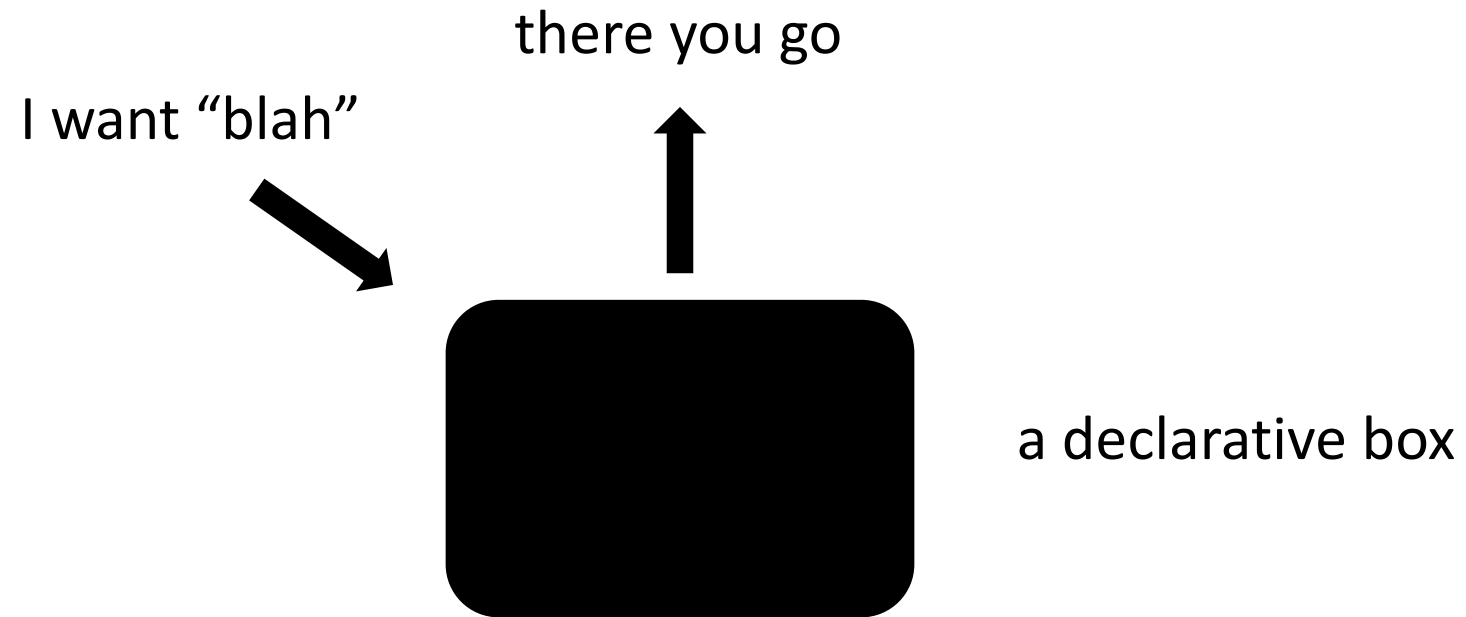
# Database Systems



why having a declarative box is useful?



# Database Systems



**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

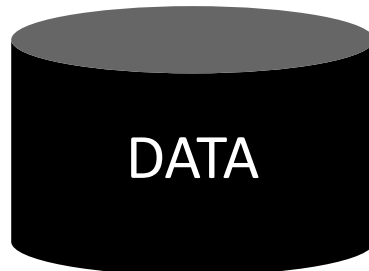
**optimization:** dynamically decide which to use

how? understand & model alternatives

# data management goals



Application





# data management goals



Application



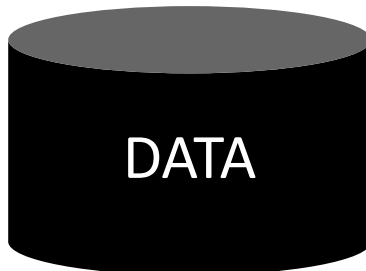
monetary cost



performance



DBMS



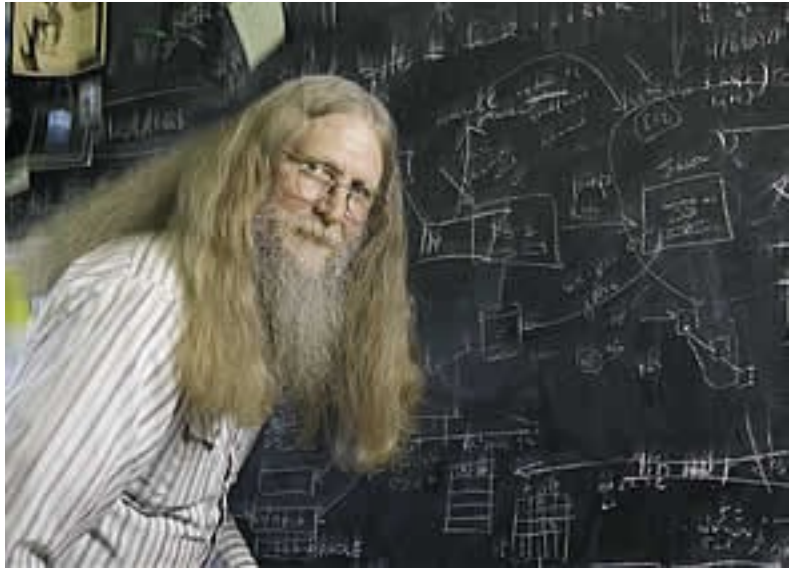
DATA



energy



hardware

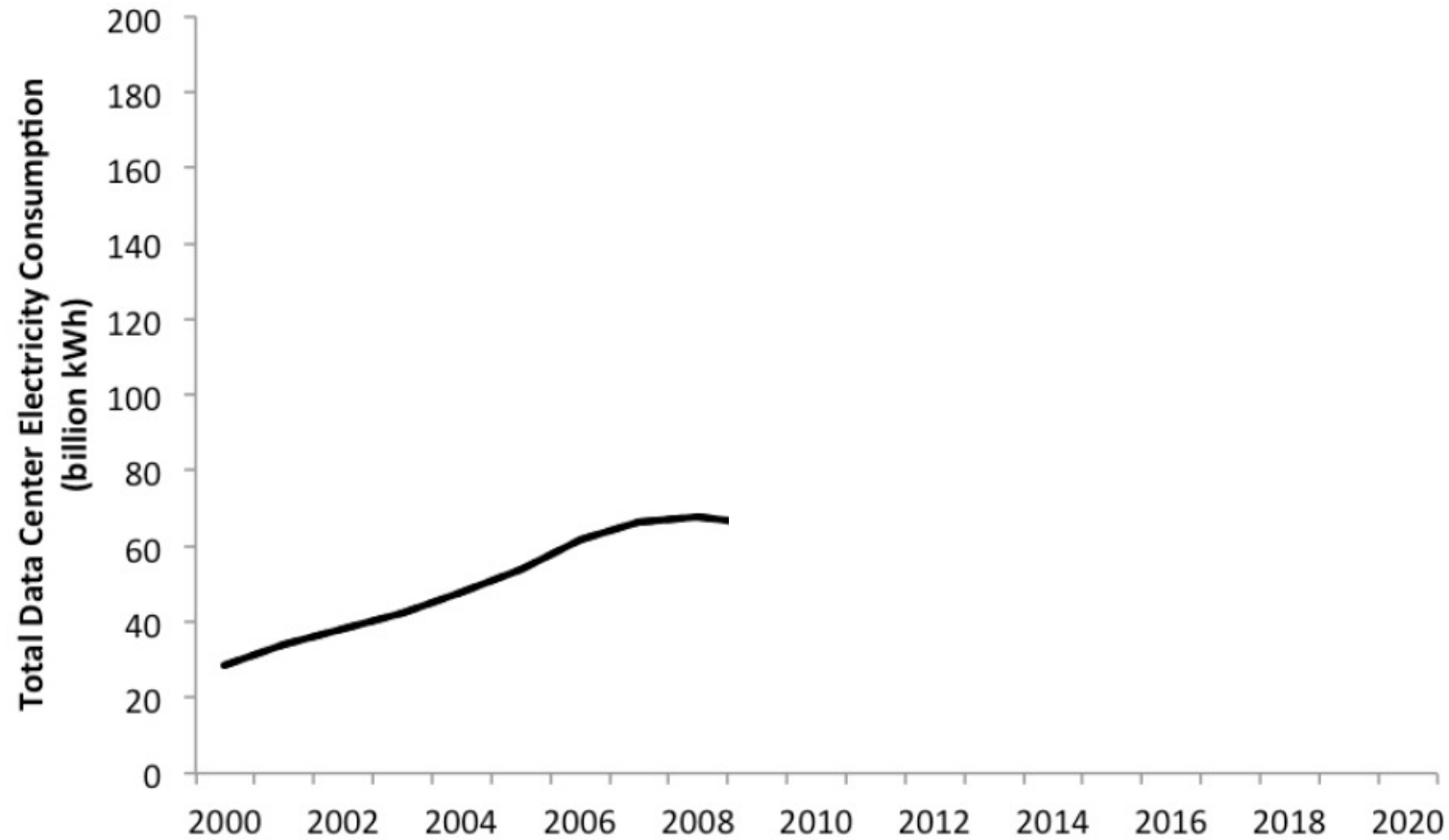


*“three things are important  
in the database world:  
**performance, performance,  
and performance**”*

Bruce Lindsay, IBM Research

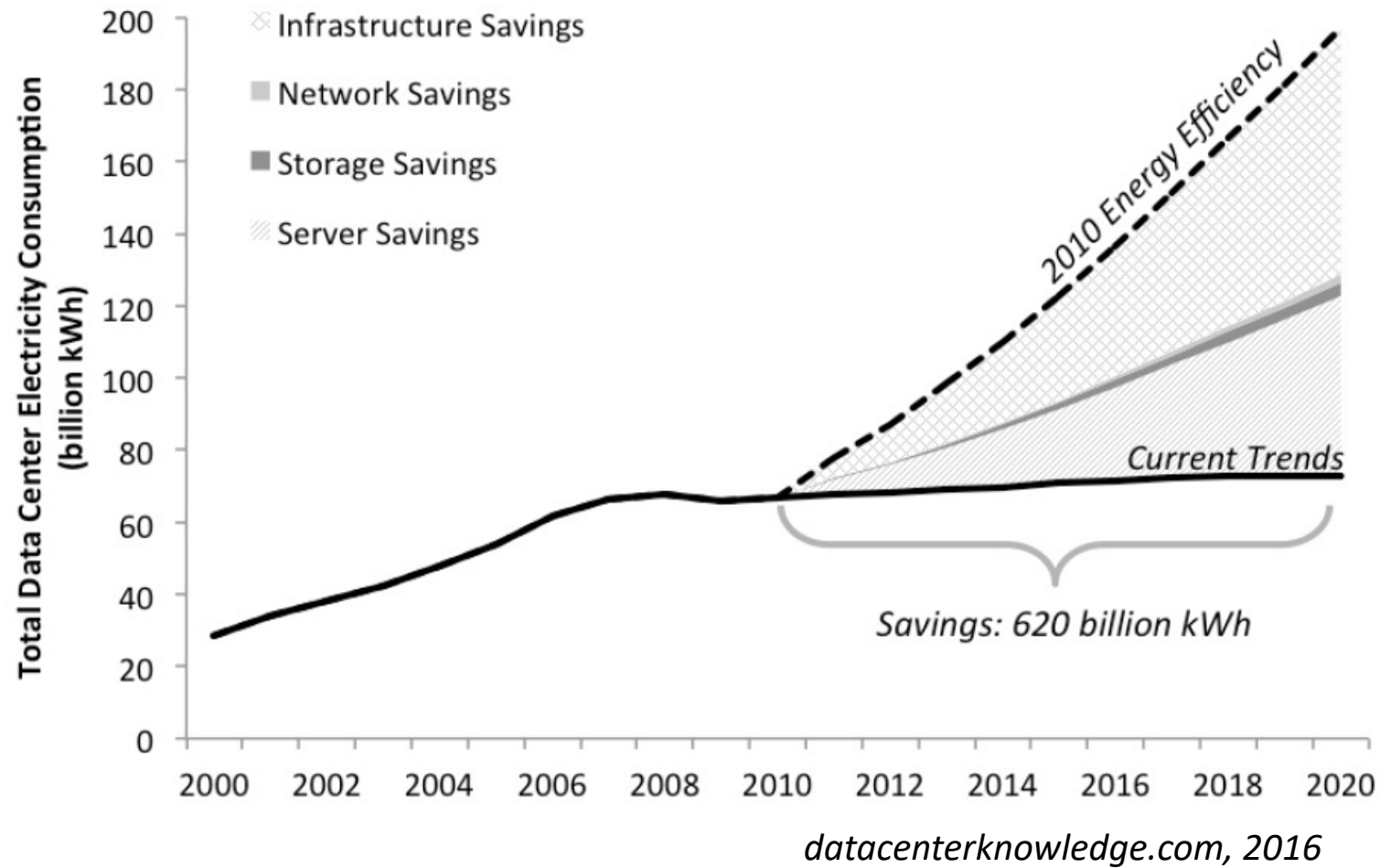
ACM SIGMOD Edgar F. Codd Innovations award 2012

# but



*datacenterknowledge.com, 2016*

# but



# but

## new hardware in the last 20 years

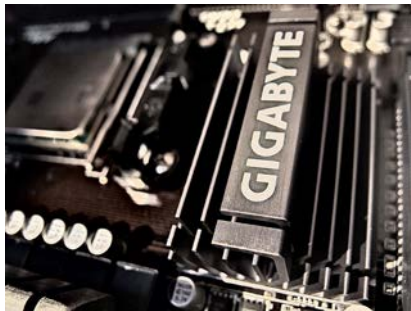
multi-core processors

multi-level cache memories

flash drives

SIMD instructions

...



# CS460

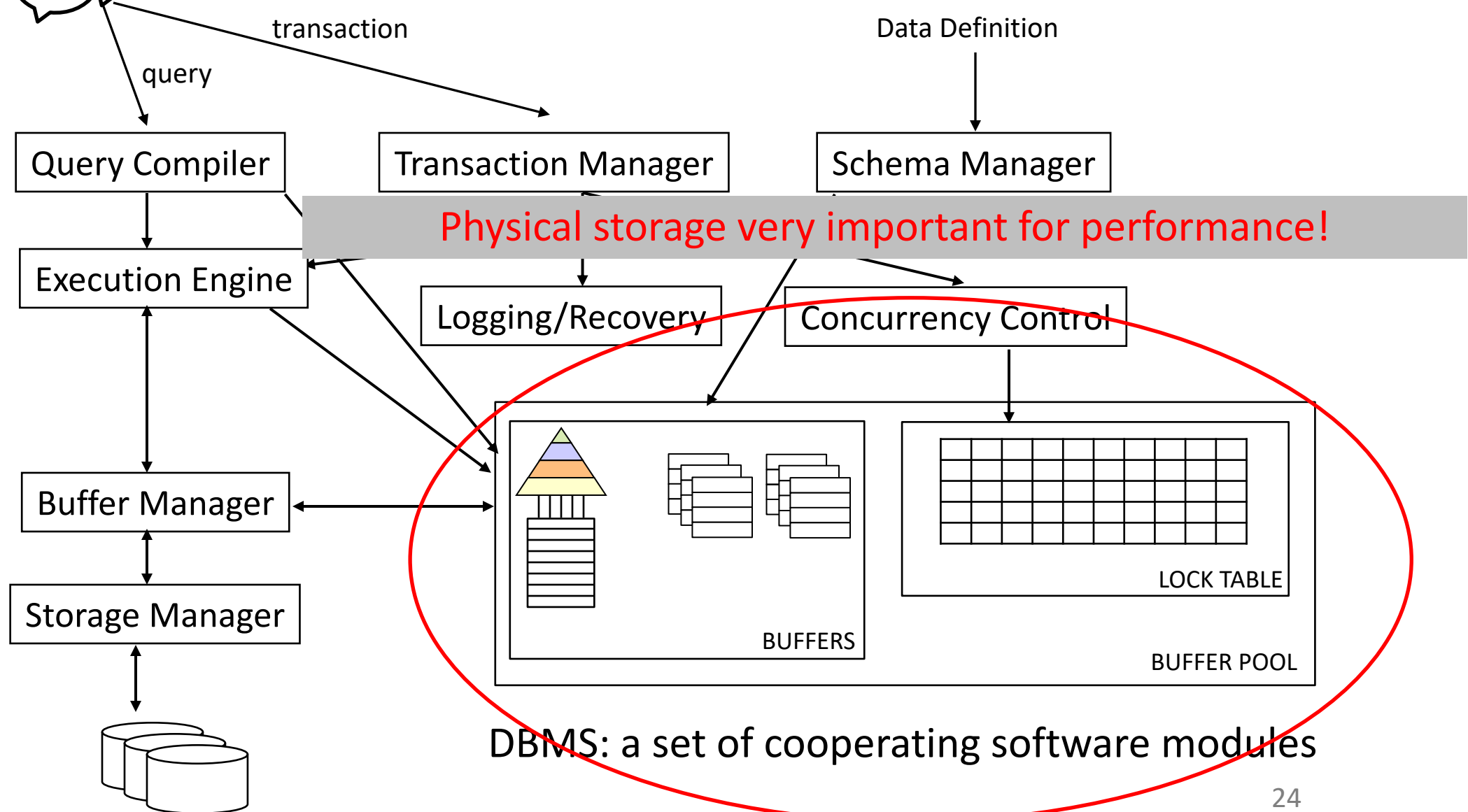
What is inside?

How it works?



performance on  
a declarative box

# Components of a "classic" DBMS



# 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?



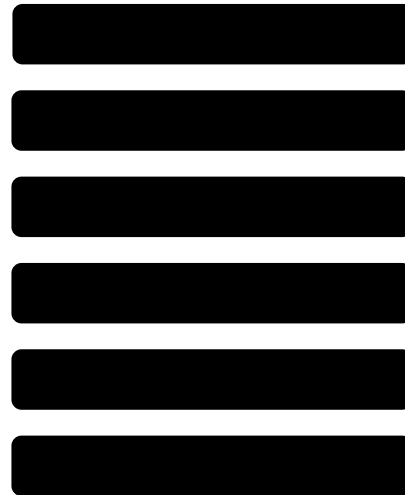
a table with rows & columns!

how to physically store it?



# how to physically store data?

one row at a time



# how to efficiently access data?



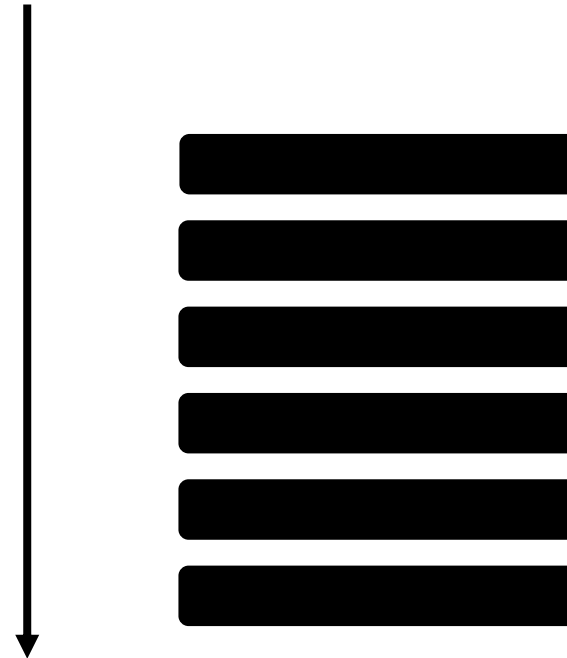
how to retrieve rows:

**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?



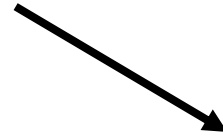
how to retrieve rows:

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



|  |
|--|
|  |
|  |
|  |
|  |
|  |
|  |

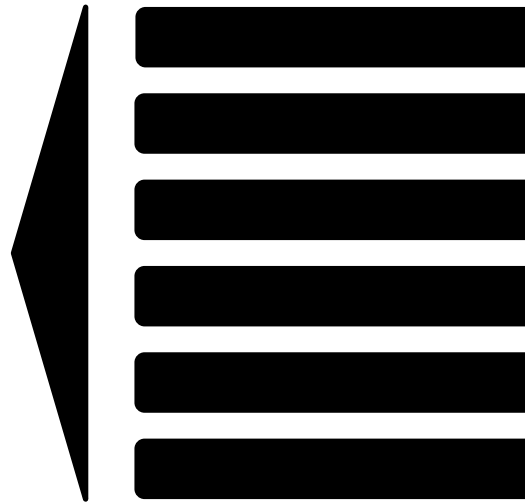
if I am interested in a single row

# 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

# 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”)



# how to efficiently access data?

## **Scan vs. Index**

Scan: many rows

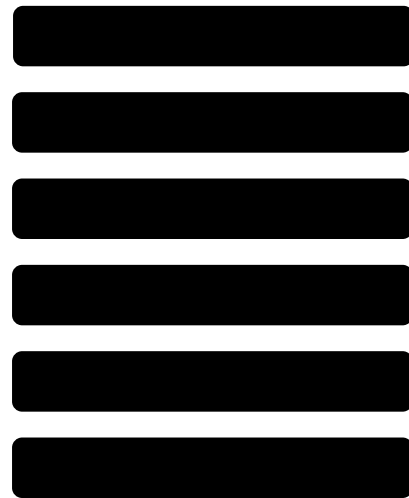
Index: few rows

# how to physically store data?

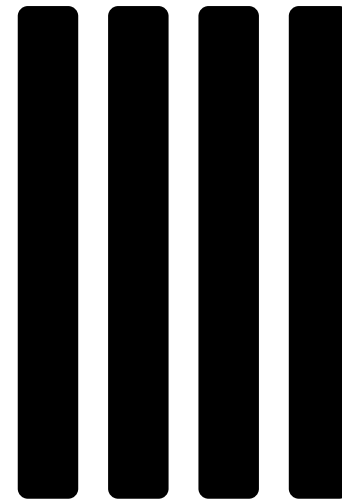
is there another way?



one row at a time

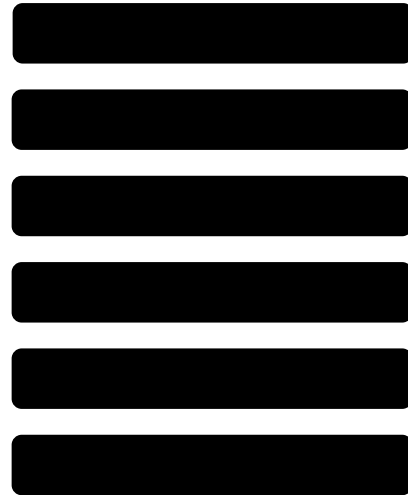


columns first

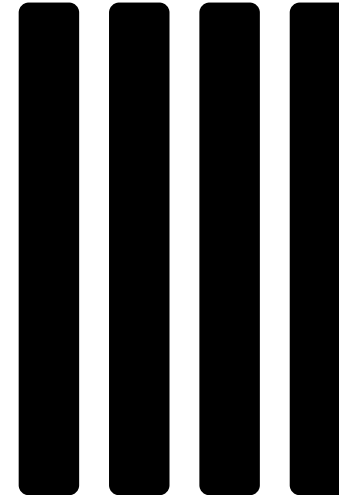


# how to efficiently access data?

rows first



columns first



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?

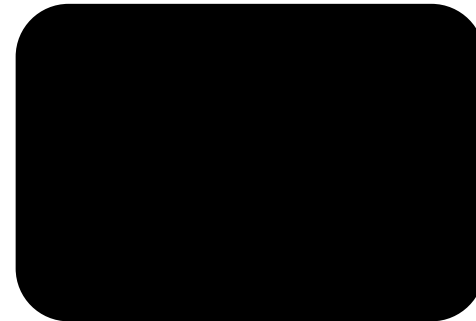
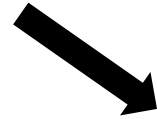
## **Rows vs. Columns**

Rows: many attributes+few rows

Columns: few attributes+lots of rows

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

I want "blah"



there you go



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?

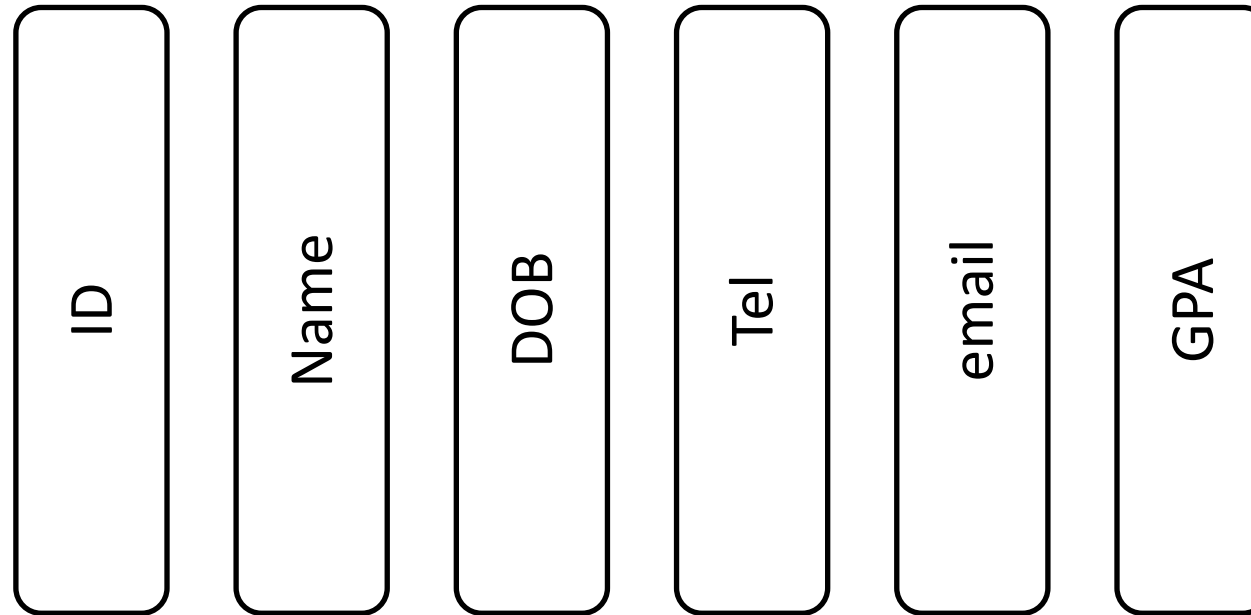
|                                     |
|-------------------------------------|
| ID   Name   DOB   Tel   email   GPA |
|-------------------------------------|

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?

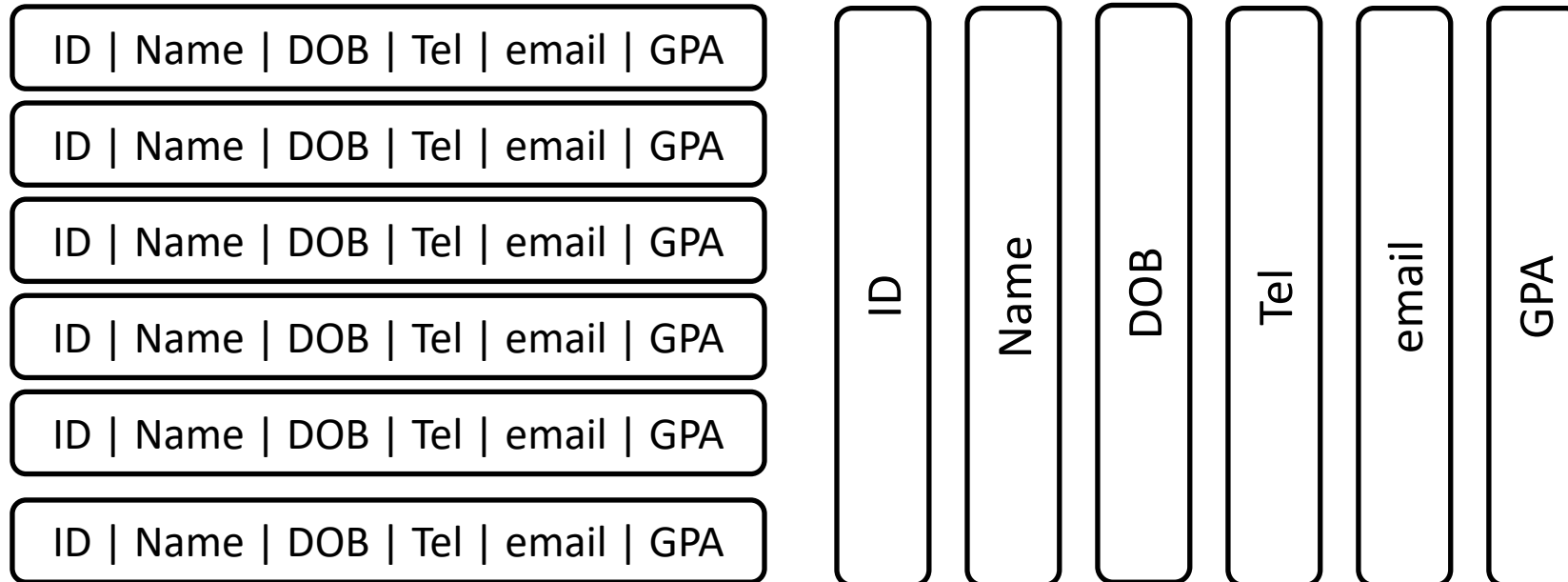


simpler basic operators

complicated query logic (more operators to connect)



does that affect the way we apply *updates*?



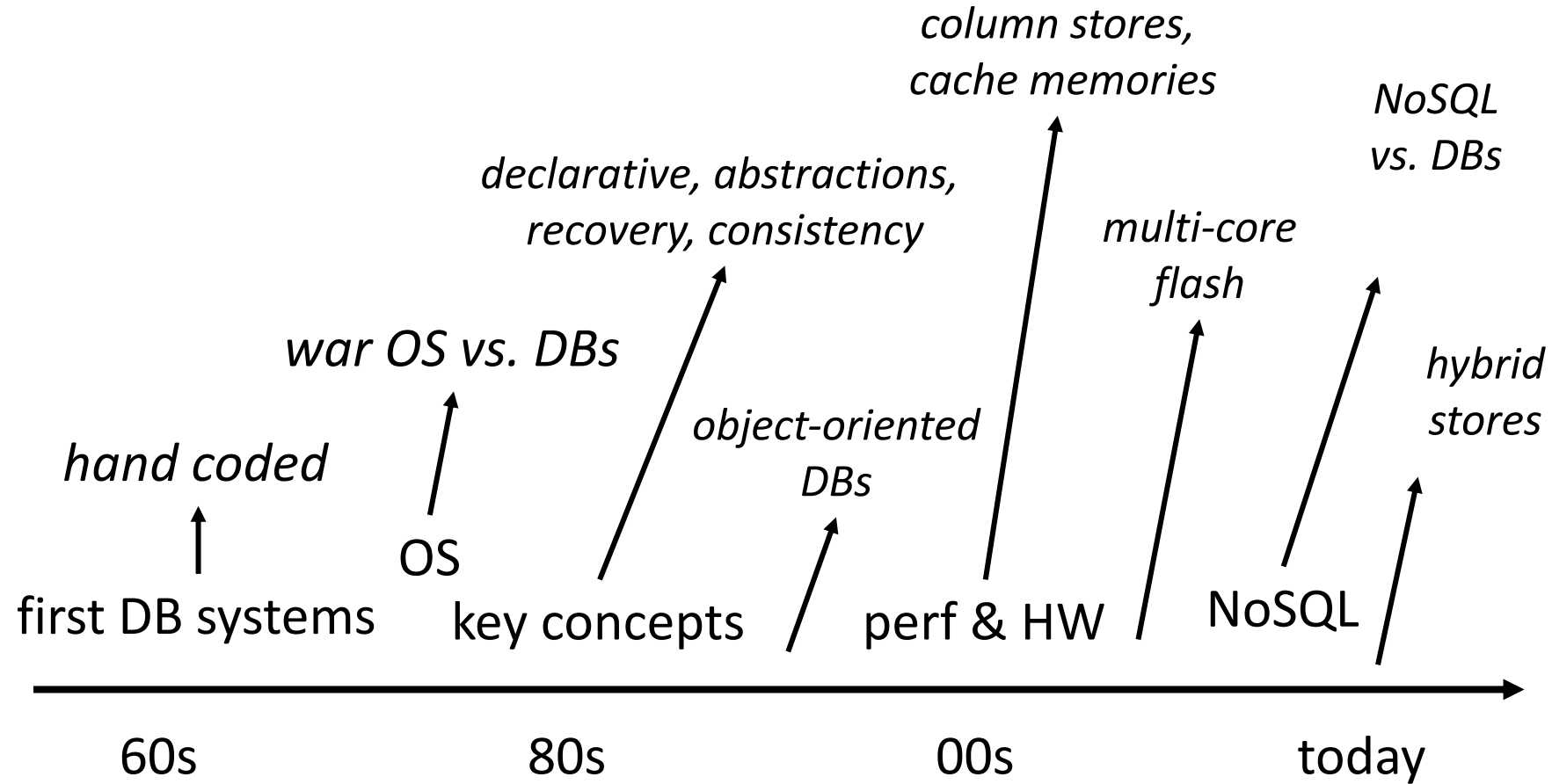
how to insert a new row?

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

## Graph Stores

natural representation of graph links

data model: **nodes & relationships**

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

# Programming Assignment 1

design, implement, & document a database application  
for movies, actors, and reviews/user likes

- (1) model and design the desired database
- (2) augment the schema to support additional functionality
  - (3) build an API to the database
  - (4) build a web app that supports:
    - (i) inserting new data, (ii) analysis queries, (iii) browsing

**project in groups of 2**

# More Programming Assignments

## *individual projects*

### **rows vs. columns**

compare the two main paradigms

### **query optimization**

understand the performance of a query

# Piazza

Announcements & Discussions in Piazza

<https://piazza.com/bu/fall2021/cs460>



# Remember & Next Time

database systems: performance (energy, HW)

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

PA1: build a database system

More programming assignments on

(i) query optimization, (ii) row-stores vs. col-stores, (ii) key-value systems

**Next: Modeling Data**