

Welcome to

CS 460: Introduction to Database Systems

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

Instructor: *Manos Athanassoulis*
email: *mathan@bu.edu*

Course Summary

We will learn how to:

- **Model data** and design good databases

ER Model, Relational

- **Query data** with SQL

- **Store & manage** data

Bits to Files to Disks, Storage Layouts, Indexes, Sorting

- **Reason** about query performance

Query evaluation & optimization

- **Update** data

Transactions, logging, ACID properties

Today

big data

data-driven world

databases & database systems



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

Big Data

buzzword ...

but ...

science / government / business / personal data

exponentially growing data collections

So, it is all good!

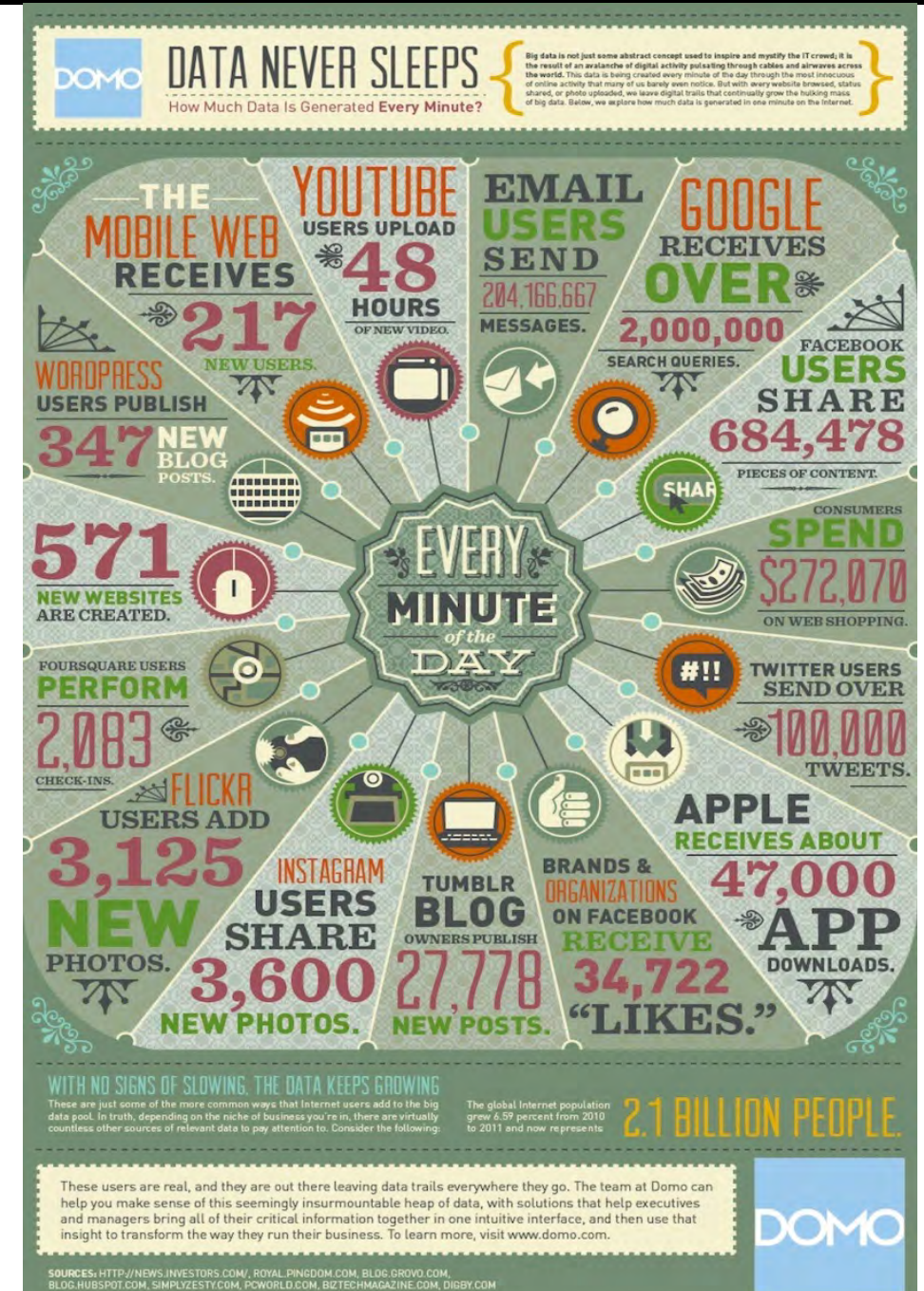
How big is “Big”?



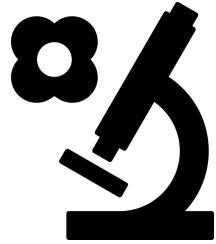
Every day, we create 2.5 exabytes* of data — 90% of the data in the world today has been created in the last two years alone.

[Understanding Big Data, IBM]

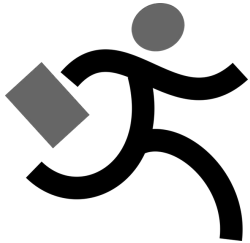
*exabyte = 10^9 GB



Using Big Data



experimental physics (IceCube, CERN)
biology
neuroscience



data mining business datasets
machine learning for corporate and consumer



data analysis for fighting crime

... are only some examples

Data-Driven World



Big Data V's

Volume

Velocity

Variety

Veracity

Information is transforming traditional business.

[“Data, data everywhere”, Economist]

CS460

we live in a ***data-driven*** world

CS460 is about the ***basics*** for
storing, using, and managing data

your lecturer (that's me!)

Manos Athanassoulis

name in greek: Μάνος Αθανασούλης

grew up in Greece

enjoys playing basketball and the sea

BSc and MSc @ University of Athens, Greece

PhD @ EPFL, Switzerland

Research Intern @ IBM Research Watson, NY

Postdoc @ Harvard University

some awards:

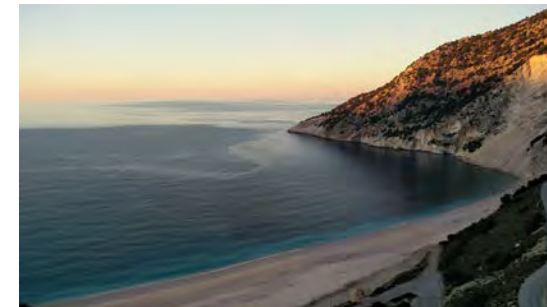
Facebook Faculty Research Award

NSF CRII Research Award

Best of SIGMOD 2017, VLDB 2017



photo for VISA / conferences



Myrtos, Kefalonia, Greece

<http://cs-people.bu.edu/mathan/>

Office: MCS 106

Office Hours: after class 3:30pm

your awesome TAs



Aneesh Raman
PhD student in DB
aneeshr@bu.edu



Tarikul Islam Papon
PhD student in DB
papon@bu.edu



Subhadeep Sarkar
Postdoc in DB
ssarkar1@bu.edu

Data

to make data usable and manageable

we organize them in collections

Databases

a large, integrated, *structured* collection of data
intended to model some real-world enterprise

Examples: a university, a company, social media

Social media: users, posts, photos

what is missing?

-- how to connect these?

-- shares, likes, friend-relationship



Database Systems

a.k.a. database management systems (DBMS)

a.k.a. data systems



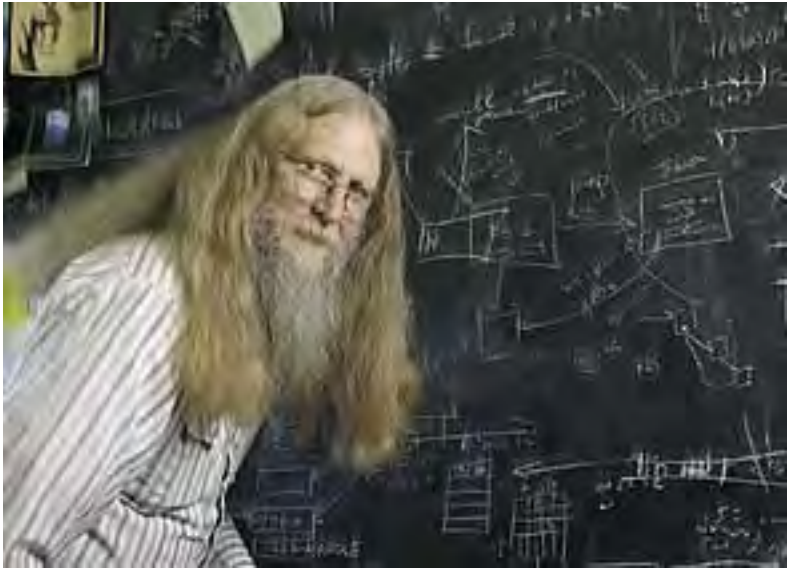
Sophisticated
pieces of software...



... which store, manage,
organize, and facilitate
access to my databases ...



... so I can do things (and ask questions) that are
otherwise hard or impossible



*“relational databases
are the foundation of
western civilization”*

Bruce Lindsay, IBM Research

ACM SIGMOD Edgar F. Codd Innovations award 2012

Ok but what really IS a database system?

Is the Internet a DBMS?



Is a File System a DBMS?



Are Social Media a DBMS?



Is the Internet a DBMS?

Not really!

Fairly sophisticated search available

web crawler *indexes* pages for fast search

.. but

data is unstructured and untyped

not well-defined “correct answer”

cannot update the data

freshness? consistency? fault tolerance?

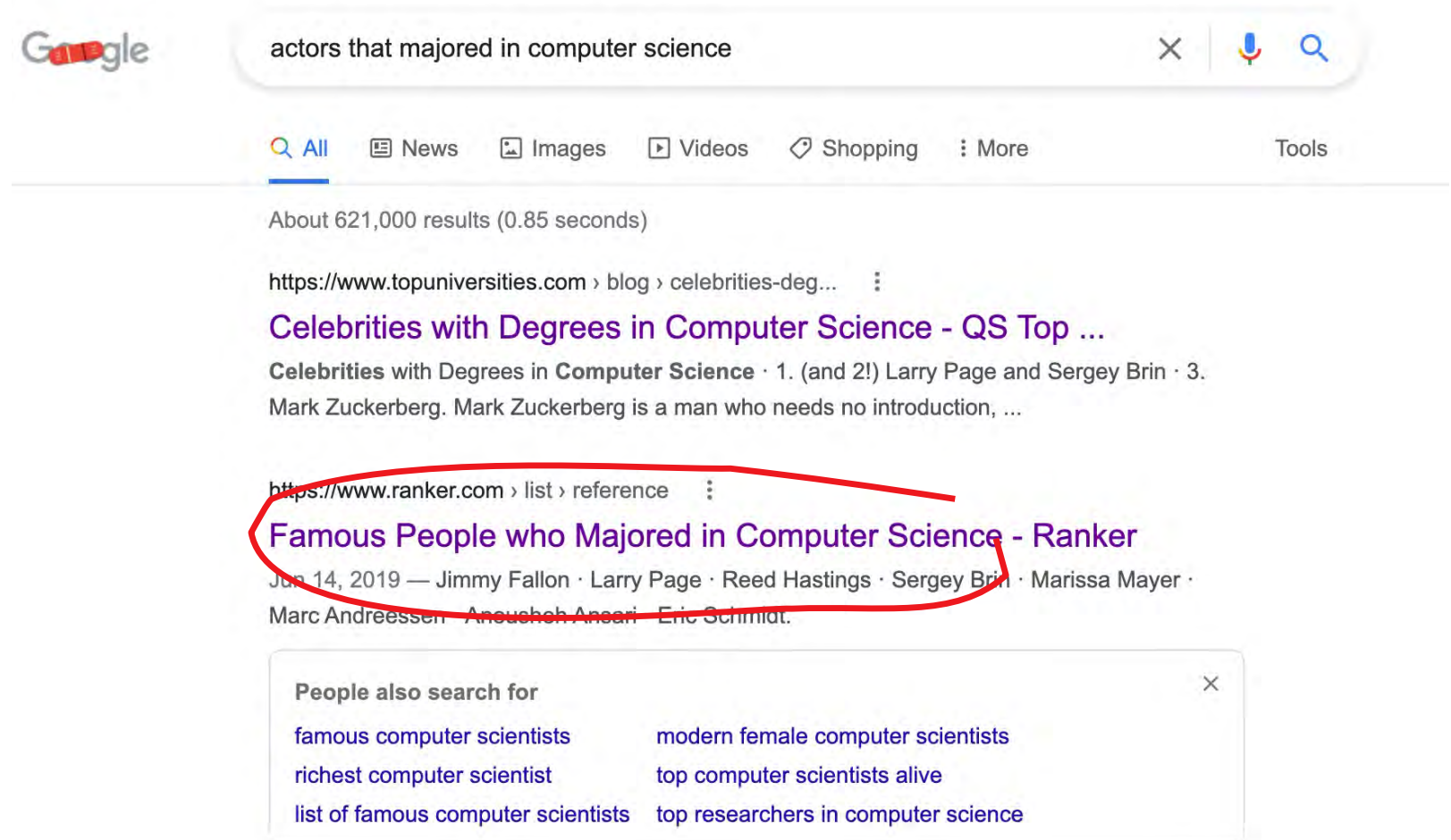
web sites **use** a *DBMS* to provide these functions

e.g., amazon.com (Oracle), facebook.com (MySQL and others)

“Search” vs. Query

What if you wanted to find out actors that have studied computer science?

Try “actors that majored in computer science” in your favorite search engine.



“Search” vs. Query

“Search” can return only what’s been “stored”

e.g., best match at Google:

The screenshot shows the Ranker website interface. At the top, there's a navigation bar with the Ranker logo and various category links: entertainment, music, nerdy, sports, living, history, culture, food, and channels. Below this is a social media sharing bar with icons for Facebook, Pinterest, and Email. The main title of the page is "Famous People who Majored in Computer Science". Underneath the title, there's a "Reference" section with a clipboard icon, stating the list was updated on June 14, 2019, with 21.6k views and 461 items. The text explains that the list is ordered by relevance and includes details like birth year and degree. It also mentions that users can filter the list by name. Below this, there's a section for "1 Jimmy Fallon" with a photo of him and a brief biography. To the right, there's an "ADVERTISEMENT" section titled "BEHIND THE VOTE" which shows other ranked individuals: Larry Page (#12 of 54 on The Most Influential CEOs of All Time), Jimmy Fallon (#13 of 69 on The Most Successful Saturday Night Live Alumni), and Eric Schmidt (#28 of ...).

Ranker
vote on >> entertainment // music // nerdy // sports // living // history // culture // food // channels

READ

Famous People who Majored in Computer Science

Reference
Updated June 14, 2019 • 21.6k views • 461 items

List of famous people who majored in computer science, including photos when available. This list of famous computer science majors is ordered loosely by relevance, meaning the most well-known people will be towards the top. The names of the colleges or universities that these prominent computer science majors attended are displayed next to each person's name- you can also see other bits of information such as what year the person was born and what kind of degree they received. If you're looking for a particular celebrity who majored in computer science you can use the "filter" bar to search for a specific name.

People here include everything from Larry Page to Reed Hastings.

This list answers the questions, "Which celebrities were computer science majors?" and "What are the names of popular people who studied computer science?"

Share this list of respected computer science majors by clicking one of the social media icons at the top or bottom of the page. Some of these people might not necessarily be actors or athletes, but they're certainly all renowned in their own line of work.

1 Jimmy Fallon

James Thomas Fallon (born September 19, 1974) is an American comedian, actor, television host, singer, writer, and producer. He is known for his work in television as a cast member on Saturday Night Live and as the host of late-night talk show The Tonight Show Starring Jimmy Fallon and before that Late Night with Jimmy Fallon. He grew up in...

BEHIND THE VOTE

Larry Page is also ranked **#12** of 54 on The Most Influential CEOs of All Time

Jimmy Fallon is also ranked **#13** of 69 on The Most Successful Saturday Night Live Alumni

Eric Schmidt is also ranked **#28** of ...

A “Database Query” Approach

where can we find accurate data for actors, universities, and majors?



A “Database Query” Approach



Males/Females (Sorted by Name Ascending)

10,220-10,269 of 6,322,145 names | [Next »](#)

Sort by: [STARMeter](#) | [A-Z▲](#) | [Birth Date](#) | [Death Date](#)



10,220. **Aaron Dow**

Production Manager | [Log](#)

Aaron Downing is a p
[Logan](#) (2017), [Walk t](#)

10,221. **Aaron Dow**

Casting Department | [Jou](#)

U.S. DEPARTMENT OF EDUCATION

OFFICE OF POSTSECONDARY EDUCATION

DAPIP

Database of Accredited
Postsecondary Institutions and Programs

1-855-831-9922

dapip@inovas.net

Help

Boston University Alumni &
Friends

**BOSTON
UNIVERSITY**

Search for School/Sites to view accreditation

“Actors” JOIN “Accredited Universities” JOIN “Alumni”
WHERE Actors.Name=Alumni.Name AND Alumni.Major=CS



Is a File System a DBMS?

Thought Experiment 1:

- You and your project partner are editing the same file.
- You both save it at the same time.
- Whose changes survive?



A) Yours **B) Partner's** **C) Both** **D) Neither** **E) ???**



Is a File System a DBMS?

Thought Experiment 1:

- You and your project partner are editing the same file.
- You both save it at the same time.
- Whose changes survive?



A) Yours **B) Partner's** **C) Both** **D) Neither** **E) ???**

Thought Experiment 2:

- You're updating a file.
- The power goes out.
- Which of your changes survive?



A) All **B) None** **C) All Since last save** **D) ???**



Is a File System a DBMS?

Not really!

Thought Experiment 1:

- You and your project partner are editing the same file.
- You both save it at the same time.
- Whose changes survive?



A) Yours **B) Partner's** **C) Both** **D) Neither** **E) ???**

Thought Experiment 2:

- You're updating a file.
- The power goes out.
- Which of your changes survive?



A) All **B) None** **C) All Since last save** **D) ???**



Are Social Media a DBMS?

Is the data structured & typed?



Not really!

Does it offer well-defined queries?

Does it offer properties like “durability” and “consistency”?

Facebook is a data-driven company that uses several database systems (>10) for different use-cases (internal or external).

Why take this class?

computation to information

corporate, personal (web), science (big data)

database systems everywhere

data-driven world, data companies

DBMS: much of CS as a practical discipline

languages, theory, OS, logic, architecture, HW

CS460 in a nutshell

model

data representation model

query

query languages – ad hoc queries

access (concurrently multiple reads/writes)
ensure *transactional* semantics

store (reliably)
maintain *consistency/semantics* in *failures*

A “free taste” of the class

data modeling

query languages

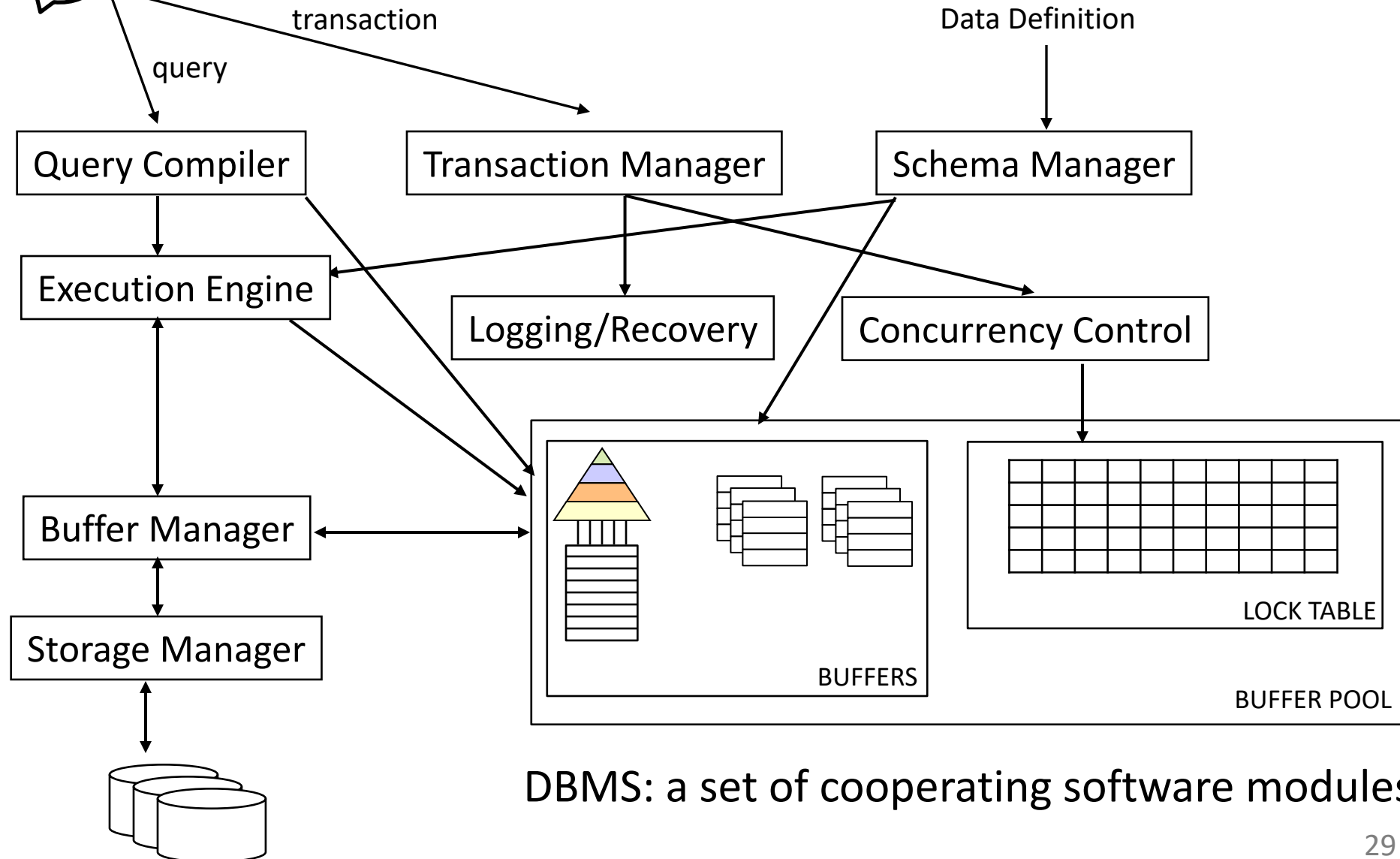
concurrent, fault-tolerant data management

DBMS architecture

Coming in next class

Discussion on *database systems designs*

Components of a "classic" DBMS



Describing Data: Data Models

data model : a collection of concepts describing data

relational model is the most widely used model today
key concepts

relation : basically a table with rows and columns

schema : describes the columns (or fields) of each table

Schema of “University” Database

Students

***sid**: string, **name**: string, **login**: string, **age**: integer, **gpa**: real*

Courses

***cid**: string, **cname**: string, **credits**: integer*

Enrolled

***sid**: string, **cid**: string, **grade**: string*



Levels of Abstraction

what the users *see*

External Schema 1

External Schema 2

what is the *data model*

Conceptual Schema

how the data is *physically* stored
e.g., files, indexes

Physical Schema

Schemas of “University” Database

Conceptual Schema

Students

***sid**: string, **name**: string, **login**: string, **age**: integer, **gpa**: real*

Courses

***cid**: string, **cname**: string, **credits**: integer*

Enrolled

***sid**: string, **cid**: string, **grade**: string*

Physical Schema

relations stored in heap files
indexes for sid/cid

External Schema

a “view” of data that can be derived from the existing data

*Course_Info (**cid**: string, **enrollment**:integer)*

*which **combines** information from Courses & Enrolled*

Data Independence

Abstraction offers “application independence”

Logical data independence

Protection from changes in *logical* structure of data

Physical data independence

Protection from changes in *physical* structure of data

Q: Why is this particularly important for DBMS?

Applications can treat DBMS as
black boxes!



Queries

”Bring me all students with gpa more than 3.0”

“SELECT * FROM Students WHERE gpa>3.0”

SQL – a powerful declarative query language

treats DBMS as a black box

What if we have multiples accesses?

Concurrency Control

multiple users/apps

Challenges



how frequent access to slow medium

how to keep CPU busy

how to avoid *short jobs* waiting behind *long ones*

e.g., ATM withdrawal while summing all balances

interleaving actions of different programs

Concurrency Control

Problems with *interleaving* actions of diff. programs



Bill



Move 100 from
savings to checking



Alice

Bad interleaving:

Savings $\text{--} = 100$

Print balances

Checking $\text{+} = 100$

Printout is missing 100\$!

Concurrency Control

Problems with *interleaving* actions of diff. programs



Bill



Move 100 from
savings to checking



Alice

What is a correct interleaving?

Savings -= 100

Checking += 100

Print balances

How to achieve this interleaving?



Scheduling Transactions

Transactions: atomic sequences of **R**eads & **W**rites

$$T_{\text{Bill}} = \{R1_{\text{Savings}}, R1_{\text{Checking}}, W1_{\text{Savings}}, W1_{\text{Checking}}\}$$
$$T_{\text{Alice}} = \{R2_{\text{Savings}}, R2_{\text{Checking}}\}$$

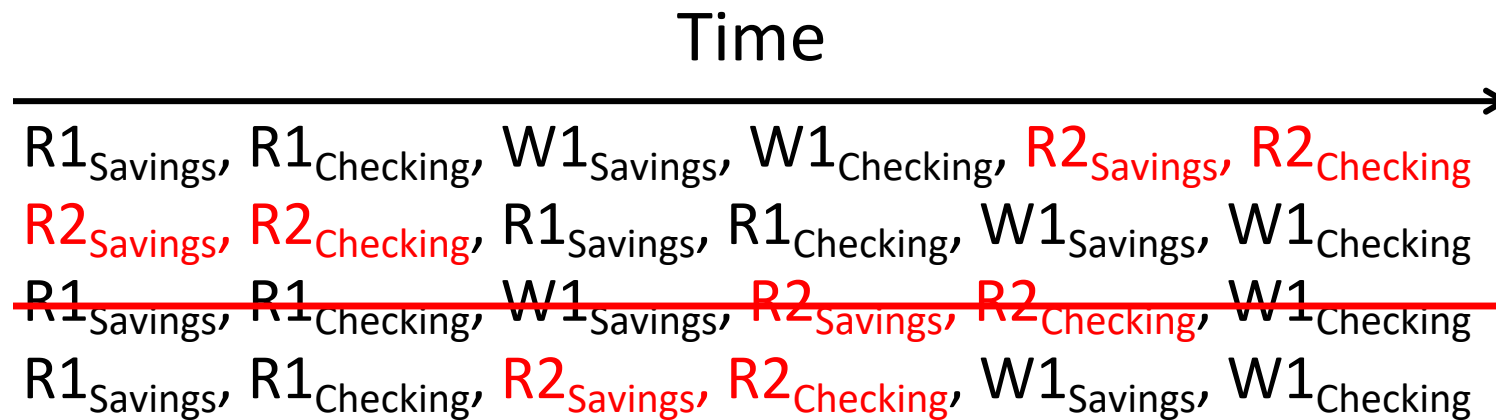
How to avoid previous problems?



Scheduling Transactions

All interleaved executions equivalent to a serial

All actions of a transaction executed as a whole



How to achieve one of these?



Locking



before an object is accessed a lock is requested

Locking



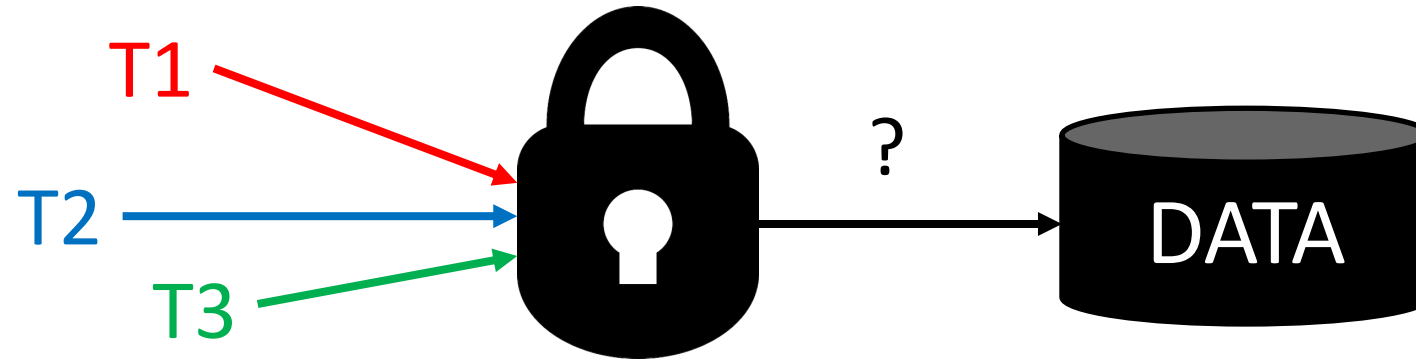
before an object is accessed a lock is requested

Locking



before an object is accessed a lock is requested

Locking



locks are held until the end of the transaction

*[this is only one way to do this, called
“strict two-phase locking”]*

Locking

$$T_1 = \{R1_{\text{Savings}}, R1_{\text{Checking}}, W1_{\text{Savings}}, W1_{\text{Checking}}\}$$
$$T_2 = \{R2_{\text{Savings}}, R2_{\text{Checking}}\}$$

Both should lock *Savings* and *Checking*

What happens:

if T1 locks Savings & Checking ?

T2 has to wait

if T1 locks Savings & T2 locks Checking ?

we have a deadlock



How to solve deadlocks?

we need a mechanism to undo

also when a transaction is incomplete
e.g., due to a crash



what can be an undo mechanism?



log every action before it is applied!

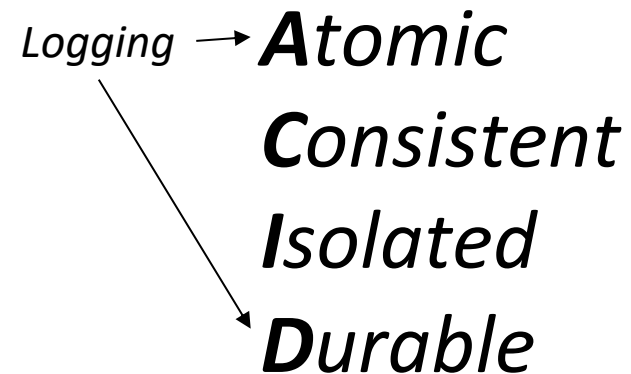
Transactional Semantics

Transaction: one execution of a user program

multiple executions → multiple transactions

Every transaction:

Logging → ***Atomic***
Consistent
Isolated
Durable

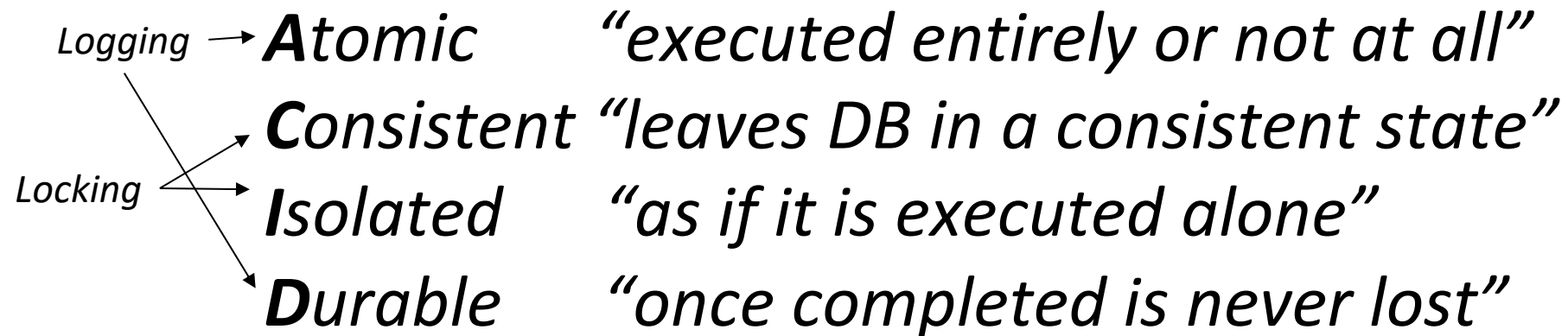


Transactional Semantics

Transaction: one execution of a user program

multiple executions → multiple transactions

Every transaction:



Who else needs transactions?



lots of data

lots of users

frequent updates

background game analytics



Scaling games to epic proportions,

by W. White, A. Demers, C. Koch, J. Gehrke and R. Rajagopalan

ACM SIGMOD International Conference on Management of Data, 2007

Only “classic” DBMS?

No, there is much more!

NoSQL & Key-Value Stores: No transactions, focus on queries

Graph Stores

Querying raw data without loading/integrating costs

Database queries in large datacenters

New hardware and storage devices

... many exciting open problems!

Course Summary

We will learn how to:

- **Model data** and design good databases

ER Model, Relational

- **Query data** with SQL

- **Store & manage** data

Bits to Files to Disks, Storage Layouts, Indexes, Sorting

- **Reason** about query performance

Query evaluation & optimization

- **Update** data

Transactions, logging, ACID properties

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

Next time in ...

CS 460: Introduction to Database Systems

Database Systems Architectures

Class administrativia

Class project administrativia

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

Additional Accommodations

If you require additional accommodations please contact the Disability & Access Services office at aslods@bu.edu or 617-353-3658 to make an appointment with a DAS representative to determine which are the appropriate accommodations for your case.

Please be aware that accommodations cannot be enacted retroactively, making timeliness a critical aspect for their provision.

You can optionally choose to disclose this information to the instructor.

