CS @CU DISTINGUISHED LECTURE SERIES 2018

Deep Learning to Learn

SEPT. 10, 2018 (MONDAY)
11:30AM-12:30PM
DAVIS AUDITORIUM
(412 CEPSR)

Pieter Abbeel (UC Berkeley)

W4111
Introduction to Databases
Fall 2018

Computer Science Department
Columbia University

Data

Data

is for serious business

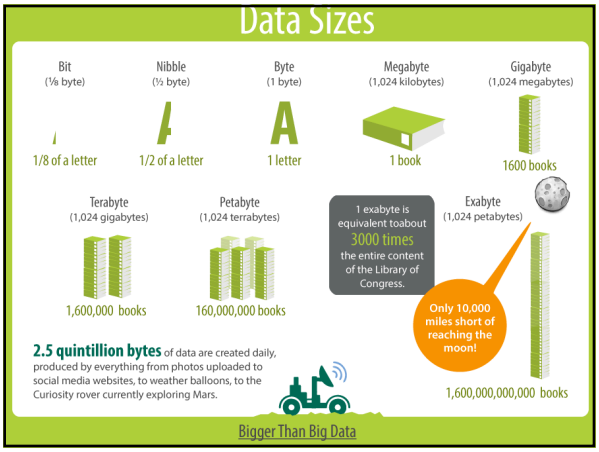
Data

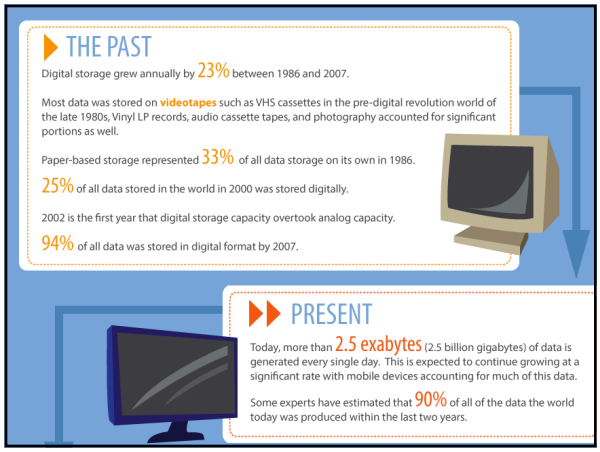
is at the center of most things.

Data

is at the center of *everything*

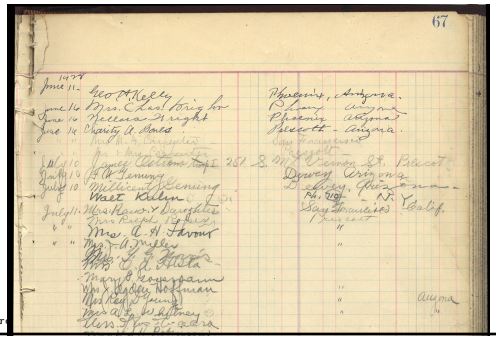
Data





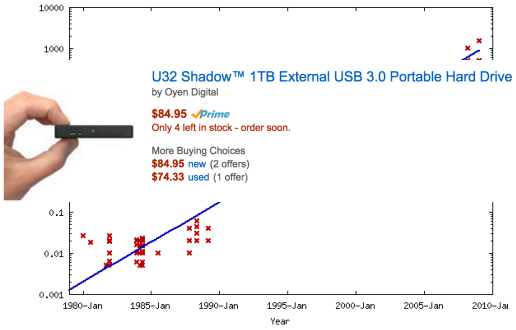
How did we get here?

Data was *Manual*



Data was *Expensive*

Data is *Cheap*



Data is *Automated*

Physical devices

Data is *Automated*

Physical devices
Software logs

Data is *Ubiquitous*

Physical devices
Software logs
Phones

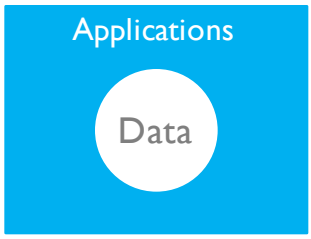
Data is *Ubiquitous*

Physical devices
Software logs
Phones
GPS/Cars

Data is *Everywhere*

Physical devices
Software logs
Phones
GPS/Cars
Internet of *Things*

What Applications?

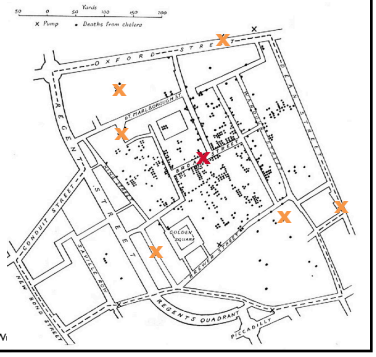


What are we doing with data?

Health

What are we doing with data?

Health



What are we doing with data?

Health

Investigative Journalism

What are we doing with data?

Health

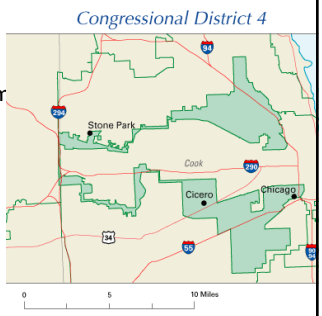
Investigative Journalism

Recommendations

What are we doing with data?

What are we doing with data?

Health
Investigative Journalism
Recommendations
Politics



What are we doing with data?

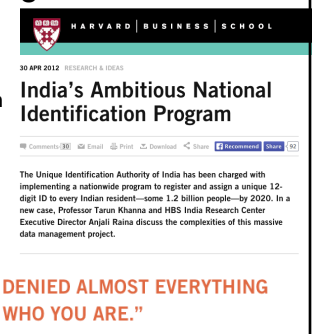
Health
Investigative Journalism
Recommendations
Politics

What are we doing with data?

Health
Investigative Journalism

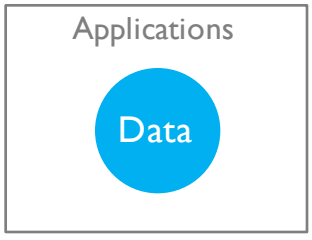
What are we doing with data?

Health
Investigative Journalism
Recommendations
Politics
Surveillance
Identity



"YOU ARE BASICALLY DENIED ALMOST EVERYTHING IF YOU CAN'T PROVE WHO YOU ARE."

What data?



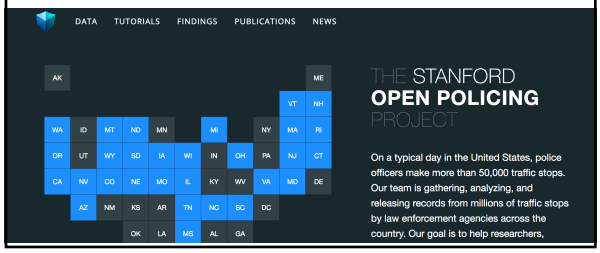
What data?

Fake data

What data?

Fake data

Biased data

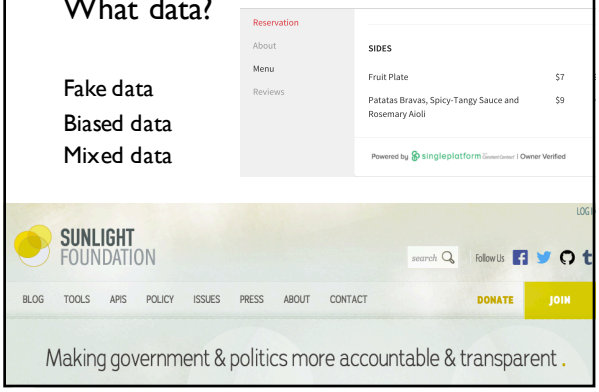


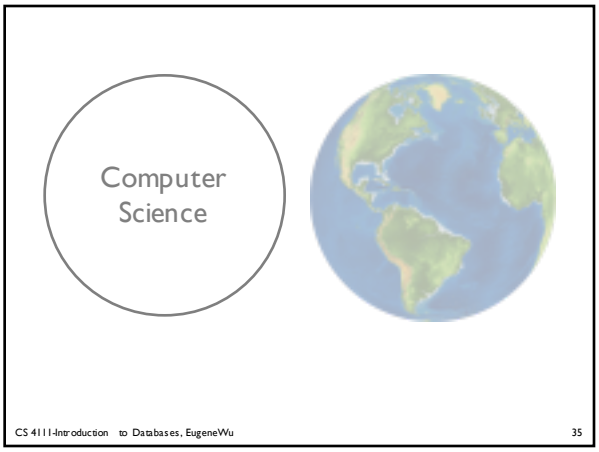
What data?

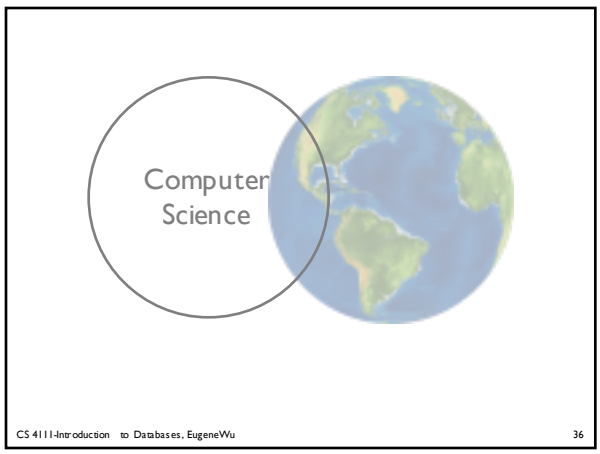
Fake data

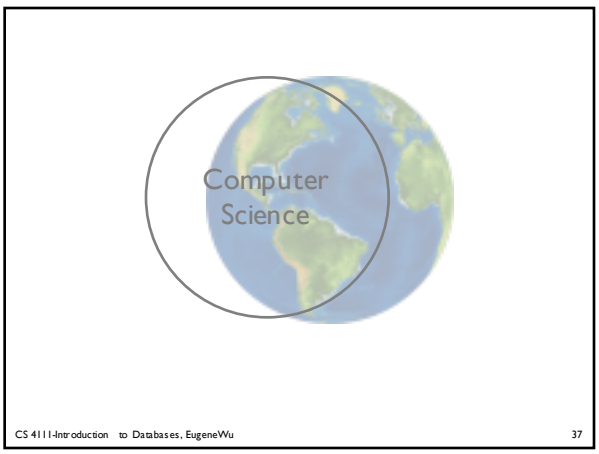
Biased data

Mixed data





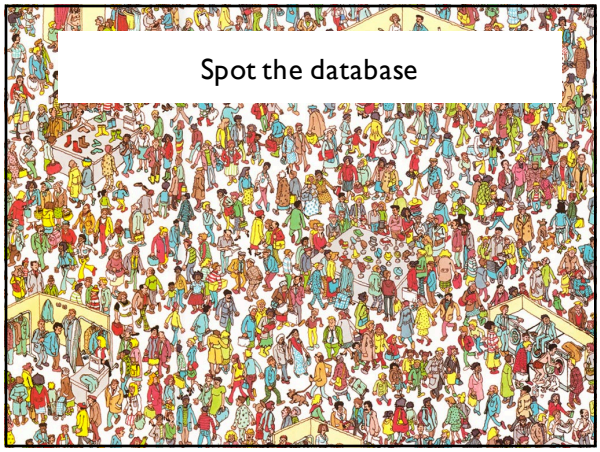




Data will be crucial to
how we live
as individuals and as a society

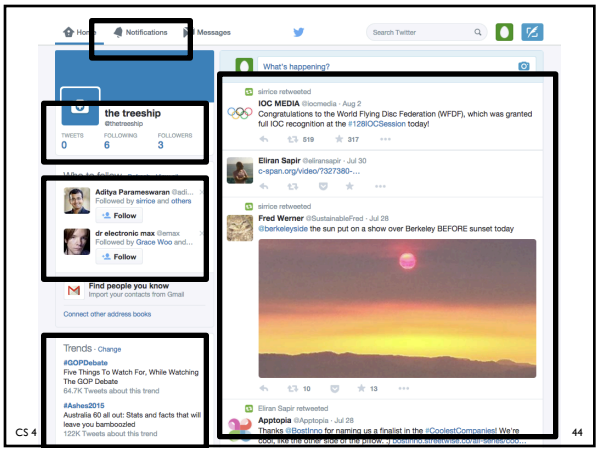
Data ~~will be~~ **is** crucial to
how we live
as individuals and as a society

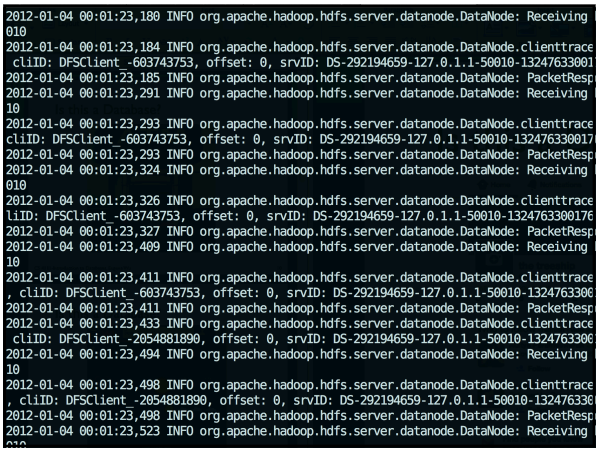
Spot the database



Google

Search Google or type URL





CS 4111-Introduction to Databases, EugeneWu 46

What is a Database?

Structured data

CS 4111-Introduction to Databases, EugeneWu 49

What is a Database?

Lots of

Structured data

CS 4111-Introduction to Databases, EugeneWu 50

Database Management System (DBMS)

A system to store, manage and access databases

CS 4111-Introduction to Databases, EugeneWu 51

Database Management System (DBMS)

System to safely and reliably store lots of persistent structured data and is convenient for multiple users to efficiently access and modify.

Is a script a DBMS?

Javascript/Python Script

- Data stored in variables (RAM)
- Very fast access

Is Excel a DBMS?

Microsoft office security
Visually access/modify/compute over data cells
Click save to store persistently

Is the file system a DBMS?

Manages files that are persistently stored on disk
Open/read/seek/write access to files
Access via file names
Access control via permissions

Is the file system a DBMS?

You and a friend edit the same text file
Save at the same time
What happens?

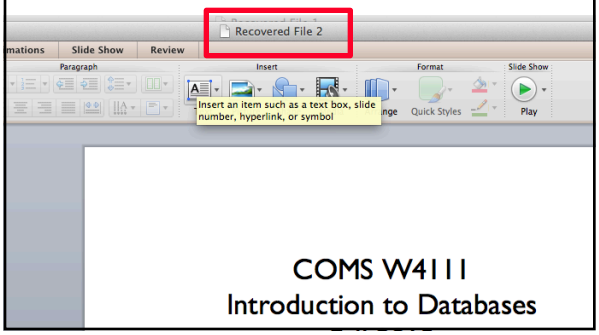
1. Your changes survive
2. Friend's changes survive
3. Both changes survive
4. No changes survive
5. ¯_(ツ)_/¯

Is the file system a DBMS?

You edit a text file
Computer crashes
What happens?

1. All changes survive
2. No changes survive
3. Changes from last save survive
4. ¯_(ツ)_/¯

Is the file system a DBMS?



COMS W4111
Introduction to Databases

Want *Guarantees* from DBMS

You want to write a hot new app on a DBMS.
What do you *not* want to worry about?

Failures disk, machine, human, corruption, deity
Lots of users
Ad-hoc data access
Data formats csv? tsv? custom format?

Database Management System (DBMS)

System to safely and reliably store lots of persistent structured data and is convenient for multiple users to efficiently access and modify.

Database Management System (DBMS)

Safe	Consistent and correct data after failures
Reliable	99.99+% Uptime
Lots	>>RAM (terabytes)
Persistent	Lives longer than DBMS application
Convenient	Physical Independence. Declarative.
Multiple Users	Concurrent access. Access control.
Efficient	*Fast*: 100k+ queries / sec

DBMSes in the Wild

Classic Relational
- $$: Oracle, IBM, Microsoft, Teradata, EMC, etc
- Free: MySQL, PostgreSQL, SQLite

New Relational
- In-Memory, Column-store, Streaming

Non-traditional
- Search (Google, Bing, Lucene), Scientific, Geo, Graph

NoSQL
- Big Data: Hadoop, Spark, etc
- Key-value: Mongo, BerkeleyDB, Cassandra, etc

DBMS-as-a-Service
- MS Azure, Google BigQuery, Amazon Redshift/RDS ...

Encompasses most of CS

OS	DBMS directly manages hardware
Languages	SQL is a domain specific language
Theory	Algorithms, models, NP-complete
AI/ML	Knowledge Discovery, KDD
Logic	Relational Algebra = 1st order logic

Scalable Computer Science

Good time to learn!

Cloud programmer
Data science
Data engineer
Machine learning engineer

DATA

Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil
FROM THE OCTOBER 2012 ISSUE

2 Key Concepts

Data Independence
Declarative Languages

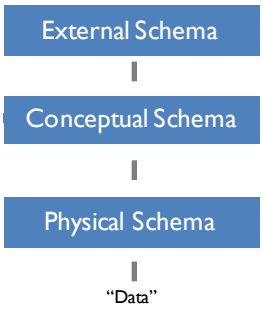
Serve to insulate application programmers from the system implementation

Data Independence

External Schema
Describe how users see data

Conceptual Schema
Describes logical structure

Physical Schema
Describes files and indexes



Example App: Guuber

Users(uid int, name str, age int)
Drivers(did int, name str)
Rides(uid int, did int, distance float, drive_time float)

Data Independence

UID	Name	Age
0	Eugene	17
1	Luis	20
2	Ken	30

0,Eugene,17
1,Luis,20
2,Ken,30

CSV File

What is the number of adults?

Data Independence

UID	Name	Age
0	Eugene	17
1	Luis	20
2	Ken	30

0,Eugene,17
1,Luis,20
2,Ken,30

CSV File

```
n = 0
for line in csv_file:
  attributes = line.split(",")
  if attributes[2] >= 18:
    n += 1
```

Data Independence

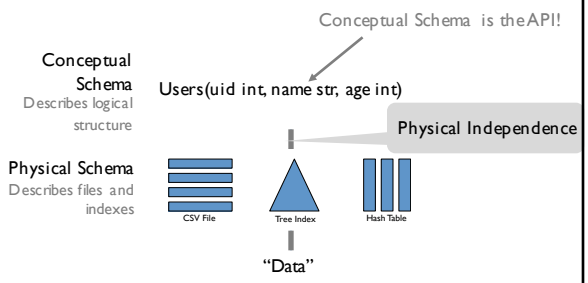
UID	Name	Age
0	Eugene	17
1	Luis	20
2	Ken	30

0,1,2
Eugene,Luis,Ken
17,20,30

CSV File

```
n = 0
For line in csv_file:
  attributes = line.split(",")
  if attributes[2] >= 18:
    n += 1
```


Data Independence



Data Independence

Users(uid int, name str, age int)
Drivers(did int, name str)
Rides(uid int, did int, distance float, drive_time float)

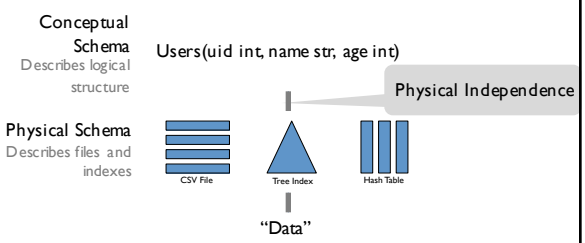
"Welcome back Mr. Wu"

Data Independence

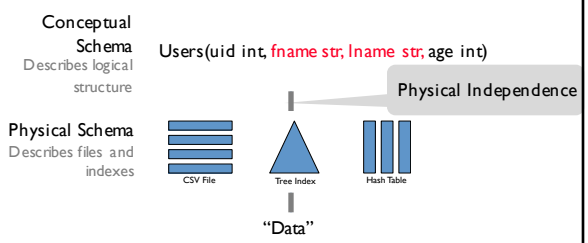
Users(uid int, fname str, lname str, age int)
Drivers(did int, name str)
Rides(uid int, did int, distance float, drive_time float)

"Welcome back Mr. Wu"

Data Independence



Data Independence

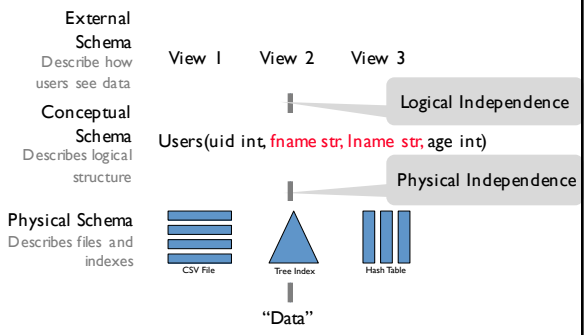


Data Independence

External Schema
Describe how users see data

Conceptual Schema
Describes logical structure

Physical Schema
Describes files and indexes



Data Independence

Physical Independence
Protection from changes in physical structure of data

Logical Independence
Protection from changes in logical structure of data

One of most important properties of a DBMS

Declarative

What you want, **not how to do it.**

"Make me a sandwich"

Buy from pb&j store
Make BLT
½ Tuna
Veggie

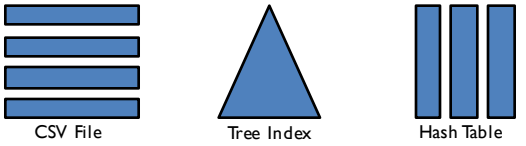
"Take two slices of wheat bread out of the 2nd shelf, put them next to each other…"

What if on 1st shelf?
Out of wheat bread?
No counter space?

Declarative

"I want all highly rated fast drivers"

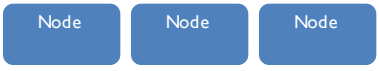
DBMS



Declarative

"I want all highly rated fast drivers"

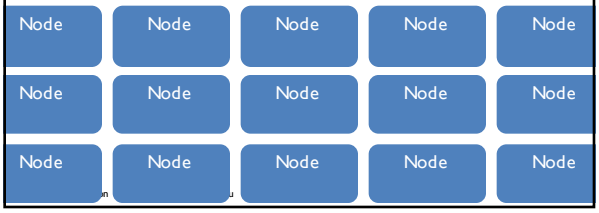
DBMS



Declarative

"I want all highly rated fast drivers"

DBMS



Classic Components in Databases

Concurrency Control
Transactions
Atomicity
Recovery and Logs

Transaction: Execution of a DB Program

Def: atomic sequence of DBMS actions

```
Begin;
<read beth's account>
<deduct from beth's account>
<increase eugene's account>
Commit;  (or Abort;)
```


Transaction: Execution of a DB Program

Def: atomic sequence of DBMS actions

Each fully executed transaction must leave DB in *consistent state* if DB is consistent before transaction

- Users specify simple *integrity constraints* on data, and DBMS enforces the constraints.
- DBMS does not understand semantics of its data
  e.g., doesn't know how bank interest is computed
- User's responsibility to ensure transaction (run alone) preserves consistency

Concurrency Control

Concurrently running multiple user programs needed for good performance
  Disk accesses are frequent & slow. Keep CPU working on several user programs while waiting.

Concurrency can cause inconsistencies
- e.g., check cleared while account balance being computed.
- *Really* hard to program against

DBMS ensures such problems don't arise
- programmers can pretend to use a single-user system.

Scheduling Concurrent Transactions

Transactions T1, …, Tn are run concurrently
Equivalent to a *serial* ordering (as if no concurrency)

Locks: T_i requests and waits for lock before read/write.
e.g., T_i locks the database, updates, then releases
e.g., T_i locks the table, updates, then releases
e.g., T_i locks rows, updates, then releases

Will talk about how this works later in course.

Atomicity

Def: Xact fully completes, or never happened
even after failures e.g., crashes

Record all actions Xact did during execution in a log

1. Write ahead logging: before making any change, ensure the change is safely recorded in log
2. After failure, read log and undo any incomplete Xacts

The Log

A log record contains enough info to undo actions:

- Transaction id
- T_i writes an object: old and new values
- Log record *must* be safely stored before the changed data
- T_i commits/aborts: store commit/abort action

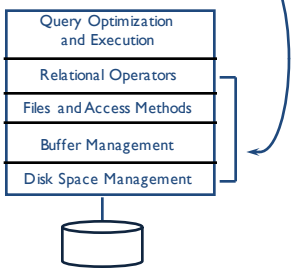
All logging, recovery and concurrency control activities hidden away from user.

Classic Structure of a DBMS

These layers must consider concurrency control and recovery

Typical layered architecture
DBMS, not OS, manages memory and disk

Doesn't show concurrency control & recovery components



Database Courses at Columbia

COMS W4111 - Intro to Databases

Prerequisites: CS3137 or CS3134; fluency in Python

Intro to DBMSes
Data Models Entity-relation, Relational, …
Relational Algebra
SQL
Applications + SQL cursors, APIs, embedded …
Normalization
Peek at DBMS internals:
- Storage and indexing
- Query optimization
- Transaction Processing

COMS W4112-Database Sys. Impl.

Prerequisites: CS3137 or CS3134; fluency in Python

Storage Methods and Indexing
Query Processing and Optimization for 1NF Relations, including external sorting
Materialized Views and Use in Query Optimization
Query Processing and Optimization for ORDBMSs
Transaction Processing and Recovery
Parallel & Distributed DBMSes: Query Proc. and Optimization
Parallel and Distributed Databases: Transaction Processing
Performance Considerations Beyond I/Os

COMS E6111-Advanced Databases

Prerequisites: CS4111; fluency in Java or Python

Information Retrieval
Web Search
Distributed Information Retrieval and Web Search
Data Mining
Data Warehousing, OLAP, Decision Support
Information Extraction
Scalable Visualization and Interaction
Supporting data analysis
Exploration, explanation and exhibition techniques

COMS E6xxx-DB Research Seminars

Prerequisites: CS4111; fluency in Java or Python

6998.002 Interactive Data Exploration Systems

6998.005 Database Topics in Research & Practice

Administrivia

Next Up

Set up your environment on the cloud

HW0 is out.
Due by Friday 9/7 10AM sharp.

Must be completed on time to stay in this class
Enrollments will be in order of completion

Your Instructor: **Eugene Wu**

B.S. @U.C. Berkeley
Ph.D. @MIT
PostDoc @U.C. Berkeley

Assistant Professor since Fall 2015

Databases, visualization, data analysis
data cleaning, crowdsourcing

Your Instructor: **Eugene Wu**

Contact

www.eugenewu.net
ewu@cs.columbia.edu
421 Mudd
1-212-939-7088

Office hours

Thurs 5:30-6:30PM
By appointment by email

Class Resources

Class web page
w4111.github.io

Discussion board
piazza (linked from website, public)

Announcements from class staff:
Website

Your TAs

Felipe Rocha
Amita Shukla
Amit Khat
Ivy Chen
Chih-Chi'an
Mayank Saxena

All TA office hours in CS TA Room (see class web page)
TA office hours will be posted on class web page

Class Information: Prerequisites

COMS W3134 - *Data Structures in Java* or
COMS W3137 - *Data Structures and Algorithms*
(equivalent courses taken elsewhere are acceptable as well)

Fluency in Python

Class Information: Lectures

Tuesdays and Thursdays
4 – 5:30PM
501 Northwest Corner Building

(here)

An aside: Success

What does succeeding in this course mean?
Timescales

How to encourage a collaborative environment?
What discourages it?

Assessment

Grading Information

Midterm 1:	25%
Midterm 2:	40% cumulative
HW:	15% (4 HWs equally weighed)
Project 1:	15%
Project 2:	5%
Extra credit:	scribe notes + advanced assignments

Median grade: B or slightly higher.
Alternative or make-up exams will not be given.
All homework assignments are equally weighted.
Project 1 has higher weight than Project 2.

Exam Dates

Midterm 1: 10/18, in class
Midterm 2: 12/6 last day of class, in class, cumulative

If you cannot make midterm 2,
do not take this course

Homework

Homeworks usually due at 10AM of due date.

No extensions or exceptions.

Three grace late days for hws throughout the semester.

After using all grace days, 25% grade deduction per late day.

Check full details on web site.

Projects (more details soon)

Two projects.

- Teams of two
- Run on cloud infrastructure
- Get CS account if your team doesn't have a computer
- Language is Python

Project 1

- Model and build your own database web application
- Explore "traditional" relational database features.

Project 2

- TBD

Projects (cont.)

No extensions or exceptions for project submission.

3 grace late days total for project.

After using all grace days, 25% grade deduction per late day.

Check full details on web site.

Extra Credit

Added *after* the curve

Scribe notes: 0-5% extra credit

Advanced Assignments:

- ~same value as HW
- Goes into more depth
- Hack on the DataBass system

Collaboration Policy

Read Syllabus on course site for allowed conduct

CS Dept academic honesty policies
http://www.cs.columbia.edu/education/honesty

We will not tolerate *any* cheating

Collaboration Policy

Discussing lectures and course material strongly encouraged

Homework and exams are *individual*. No exceptions
Any libraries or code however minor must be disclosed.

Projects are done in teams; no collaboration between teams.

Contact the instructor right away if you have any questions or are falling behind.

Textbook

Raghu Ramakrishnan, Johannes Gehrke: *Database Management Systems*, 3rd edition, McGraw-Hill, 2002

Available from

- *Bookculture bookstore 536 W. 112th St.*
- *Online retailers*
- *Upperclass-persons*
- *On reserve in Engineering Library*

Contests and Rewards

Project 1 contest

- Four best projects chosen as contest winners.
- Winners get:
  - Option to discuss and demo your project in class.
  - 10% boost in Project 1 grade.

On-going Feedback

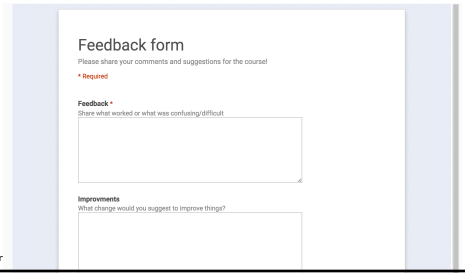
Please provide feedback throughout the course.

- What is useful or confusing in lecture
- Thoughts about software stack
- Thoughts about assignments

Email me, come to office hours, talk to staff or:

On-going Feedback

Use form on website



Slides borrow material from
Prof. Gravano
Prof. Hellerstein & Franklin@Cal
Prof. Madden & Stonebraker@MIT

(and by transitivity Raghu Ramakrishnan and Johannes Gehrke)

CS@CU DISTINGUISHED LECTURE SERIES 2018

Deep Learning to Learn

SEPT. 10, 2018 (MONDAY)
11:30AM-12:30PM
DAVIS AUDITORIUM
(412 CEPSR)

Pieter Abbeel (UC Berkeley)

w4111.github.io
ewu@cs.columbia.edu

DO HOMEWORK 0!