Introduction

Introduction to Databases

CompSci 316 Spring 2020

DUKE
COMPUTER SCIENCE

Welcome to

CompSci 316: Introduction to Database Systems!! Spring 2020

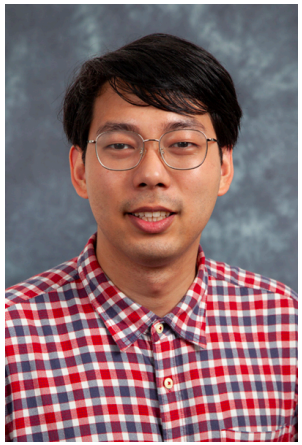
About us: instructor

- Instructor: Sudeepa Roy
  - At Duke CS since Fall 2015
  - Member of "Duke Database Devils" a.k.a. the database research group
  - PhD. UPenn, Postdoc: U. of Washington
  - Research interests:
    - "data"
    - data management, database theory, data analysis, data science, causality and explanations, uncertain data, data provenance, crowdsourcing, ….

Meet your grad TAs

Tiangang Chen

- Duke CS MS student
- Interested in computer systems
- Runs a half-marathon every year!



Xiangchen Shen

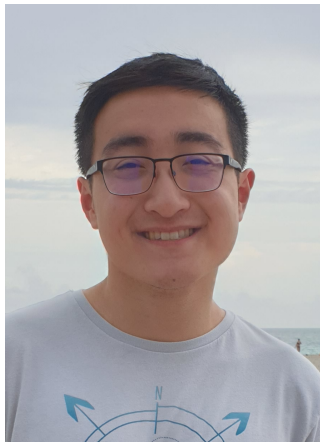
- Duke CS MS student
- Interested in data science and machine learning, currently working on image processing
- Loves cats!

Yuchao Tao

- Duke CS PhD student
- Interested in databases and privacy research
- Enjoys cooking!

* All CompSci 516 veterans

Meet your UTAs



- Duke CS major
- Interested in applying computational models and algorithms to biological systems
- Loves to downhill ski!

David Chen

- Duke CS major, minor in Visual Media studies
- Interested in UI/UX, front-end development, and project management
- Has two dogs and a hamster!

Jane Li

- Duke CS major
- Interested in coding, machine learning, and theory
- Loves hiking and bubble tea!

Runxin (Rebecca) Wang

* All CompSci 316 veterans

What are the goals of this course?

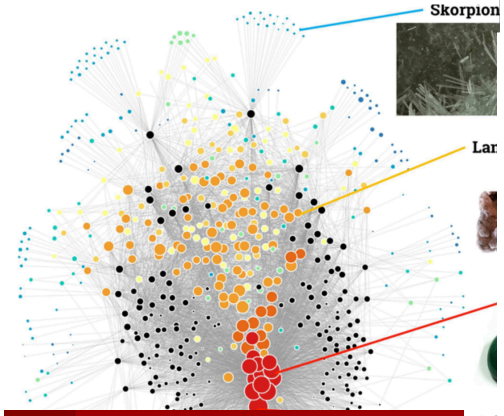
- Learn about “databases” or data management

Why do we care about data? (easy)

How big data can help find new mineral deposits

Valentina Ruiz Leotaud | Aug. 2, 2018, 4:11 PM |

PEOPLEMINE | FACEBOOK | LINKEDIN | TWITTER | EMAIL | PRINT



The New York Times

When Sports Betting Is Legal, the Value of Game Data Soars

A trader working at William Hill, an international sports betting book, in Las Vegas.

Bridget Bennett for The New York Times

Sections | The Washington Post | Try 1 month for $1 | Sign In

POLITICO

Cambridge Analytica whistleblower Chris Wylie speaks during a press conference at the Frontline Club on March 26, 2018 in London | Dan Kitwood/Getty Images

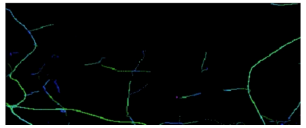
Cambridge Analytica helped 'cheat' Brexit vote and US election, claims whistleblower

Giving evidence to MPs, Chris Wylie claimed the company's actions during the Brexit campaign were 'a breach of the law.'

By MARK SCOTT | 3/27/18, 5:46 PM CET | Updated 3/29/18, 9:18 PM CET

Transportation

Researchers are trying to teach computers to forecast traffic like the weather



… The three years of gathering and analyzing data culminated in what U.S. Sailing calls their "Rio Weather Playbook," a body of critical information about each of the seven courses only available to the U.S. team…

— FiveThirtyEight, "**Will Data Help U.S. ling Get Back On The Olympic Podium?**" Aug 15, 2016

Data =

Money

Information

Power

Fun

in

Science, Business, Politics, Security Sports, Education, ….

Wait.. don't we need to take a Machine Learning or Stat course for those things?

Yes, but..

Pic: https://www.technobuffalo.com/sites/technobuffalo.com/files/styles/xlarge/public/wp/2012/05/confused-student.jpg

... we also need to manage this (huge or not-so-huge) data!

Also think about building a new App or website based on data from scratch

- E.g., your own version of book purchase platform (like a mini-Amazon)
- Large data! (think about all books in the world or even in English)

- How do we start?

* You are going to do something similar in the course project!

Who are the key people?

Who are the key people?

- At least two types:
  - Database admin (assuming they own all copies of all the books)
  - Users who purchase books
  - Let's proceed with these two only

- Other people:
  - Sellers
  - HR
  - Finance
  - Who deal with the warehouse of the books
  - ….

What should the user be able to do?

- i.e. what the interface look like? (think about Amazon)

What should the user be able to do?

- i.e. what the interface look like? (think about Amazon)

1. Search for books
   - With author, title, topic, price range, ….
2. Purchase books
3. Bookmark/add to wishlist

What should the platform do?

What should the platform do?

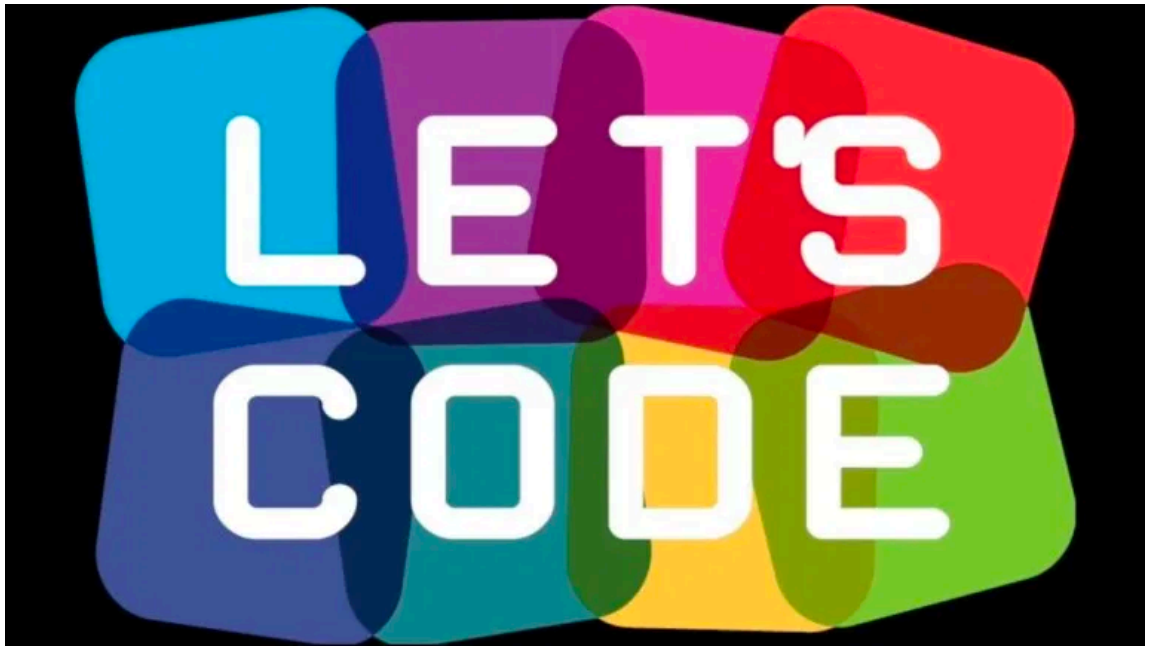
1. Returns books as searched by the authors
2. Check that the payment method is valid
3. Update no. of copies as books are sold
4. Manage total money it has
5. Add new books as they are published
6. ….

What are the desired and necessary properties of the platform?

What are the desired and necessary properties of the platform?

- Should be able to handle a large amount of data
- Should be efficient and easy to use (e.g., search with authors as well as title)
- If there is a crash or loss of power, information should not be lost or inconsistent
  - Imagine a user was in the middle of a transaction when a crash happened, paid the money, but the book has not been purchased
- No surprises with multiple users logged in at the same time
  - Imagine one last copy of a book that two users are trying to purchase at the same time
- Easy to update and program
  - For the admin

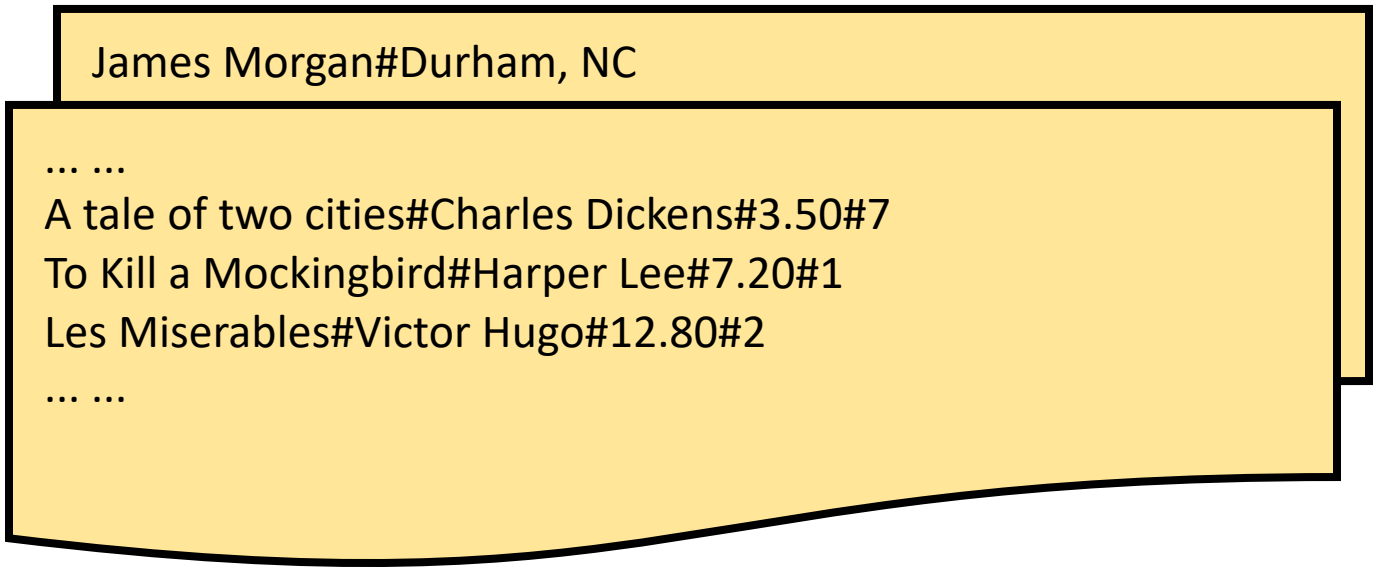
That was the design phase (a basic one though)



How about C++, Java, or Python?
On data stored in large files

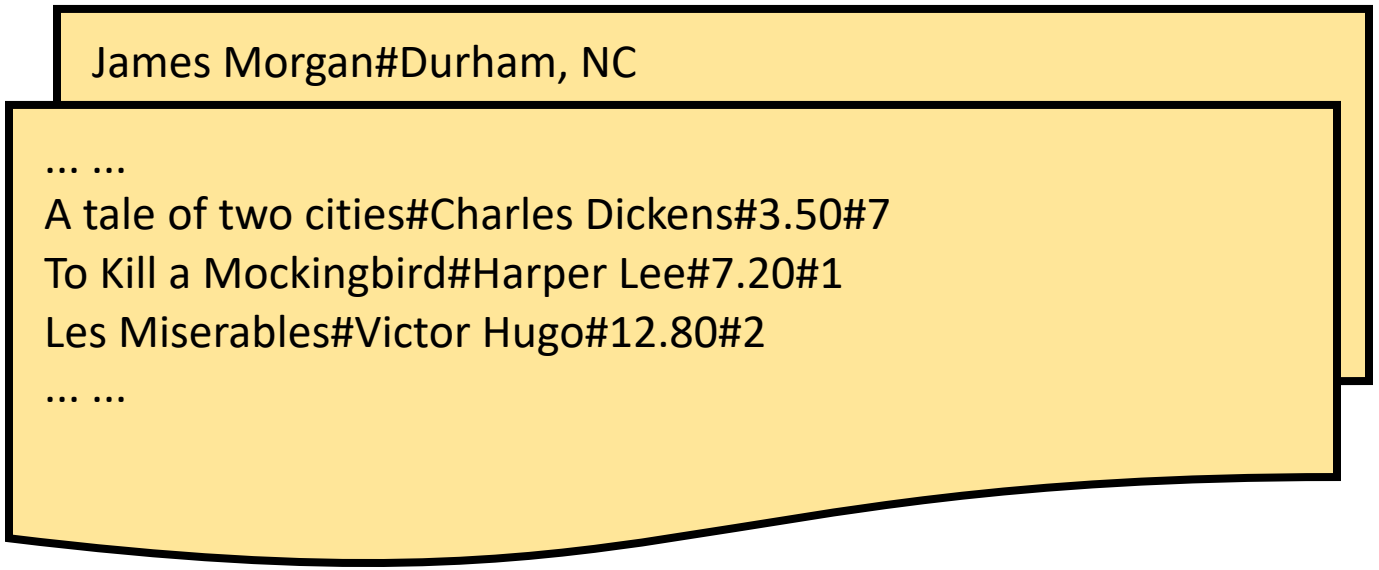
https://i1.wp.com/dynamiclandscapes.vita-learn.org/wp-content/uploads/2019/05/Lets-code.jpg?resize=768%2C432&ssl=1

Sounds simple!



- Text files – for books, customer, …
- Books listed with title, author, price, and no. of copies
- Fields separated by #'s

Query by programming



- James Morgan wants to buy "To Kill a Mockingbird"
- A simple script
  - Scan through the books file
  - Look for the line containing "To Kill a Mockingbird"
  - Check if the no. of copies is >= 1
  - Bill James $7.20 and reduce the no. of copies by 1

Better idea than scanning?

Binary search! Keep file sorted on titles

What if he changes the "query" and wants to buy a book by Victor Hugo?

Revisit: What are the desired and necessary properties of the platform?

- Should be able to handle a large amount of data
- Should be efficient and easy to use (e.g., search with authors as well as title)
- If there is a crash or loss of power, information should not be lost or inconsistent
  - Imagine a user was in the middle of a transaction when a crash happened, paid the money, but the book has not been purchased
- No surprises with multiple users logged in at the same time
  - Imagine one last copy of a book that two users are trying to purchase at the same time
- Easy to update and program
  - For the admin

Try to open a 10-100 GB file

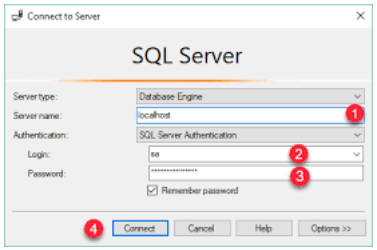
Try to search both on a large flat file

Imagine programmer's task

Imagine adding a new book or updating Copies (+ allow search) on a 10-100 GB text file

Solution?

- DBMS = Database Management System



PostgreSQL

MySQL

ORACLE DATABASE

IBM DB2

mongoDB.

A DBMS takes care of all of the following (and more):

In an easy-to-code, efficient, and robust way

- Should be able to handle a large amount of data
- Should be efficient and easy to use (e.g., search with authors as well as title)
- If there is a crash or loss of power, information should not be lost or inconsistent
  - E.g., user was in the middle of a transaction when a crash happened, paid the money, but the book has not been purchased
- No surprises with multiple users logged in at the same time
  - Imagine one last copy of a book that two users are trying to purchase at the same time
- Easy to update and program
  - For the admin

Optimization

Index

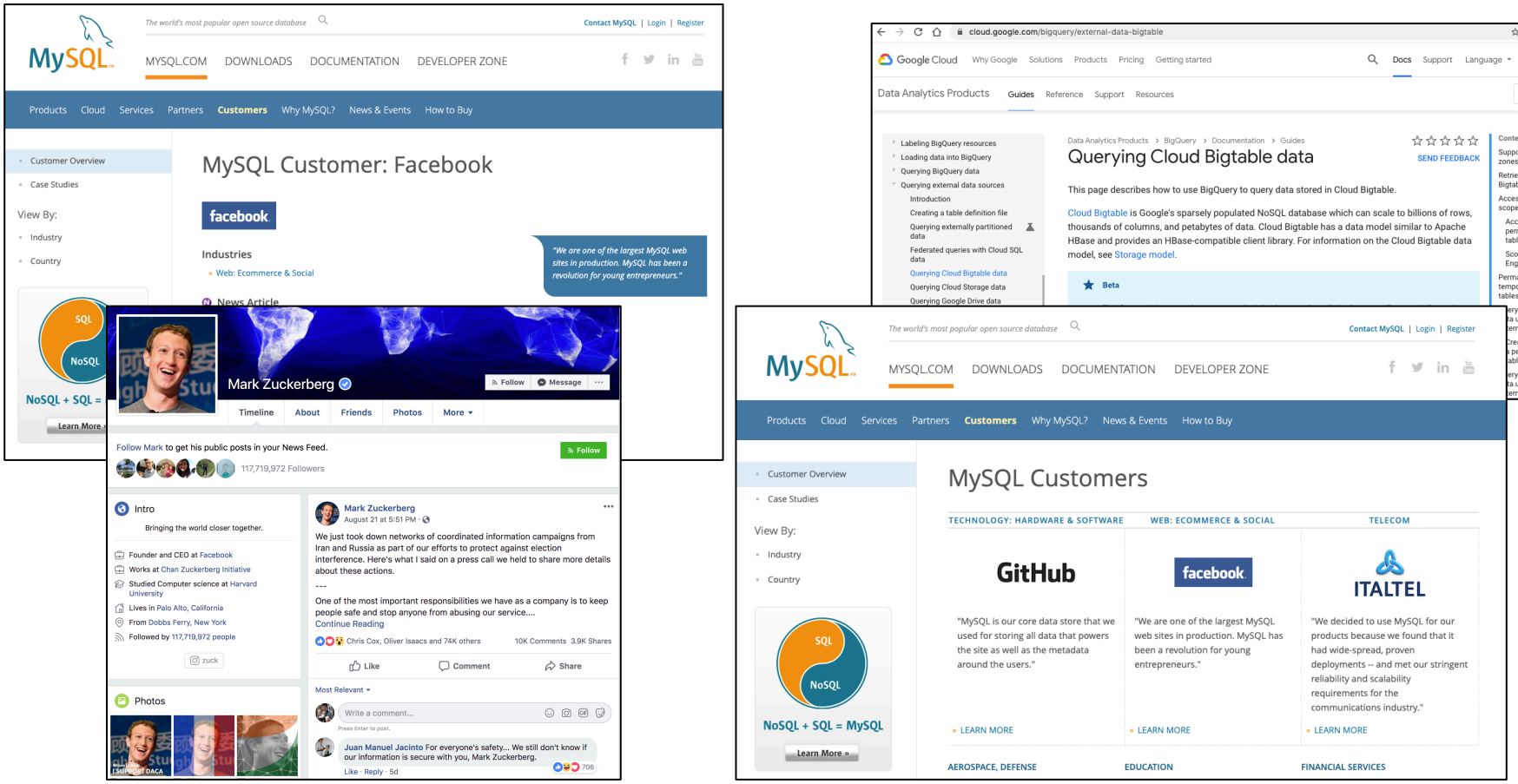
Recovery

Consistency

Declarative

* We will learn these in the course!

DBMS helps the big ones!



Note: Not always the "standard" DBMS (called Relational DBMS), but we need to know pros and cons of all alternatives

CompSci 316 gives an intro to DBMS

- How can a user use a DBMS (programmer's/designer's perspective)
  - Run queries, update data (SQL, Relational Algebra)
  - Design a good database (ER diagram, normalization)
  - Use different types of data (Mostly relational, also XML/JSON)
- How does a DBMS work (system's or admin's perspective)
  - Storage, index
  - Query processing, join algorithms, query optimizations
  - Transactions: recovery and concurrency control
- Glimpse of advance topics and other DBMS
  - NOSQL, Spark (big data)
  - Data mining
- Hands-on experience in class projects by building an end-to-end website or an app that runs on a database

Misc. course info

- Website: https://www2.cs.duke.edu/courses/spring20/compsci316/
  - Course info; tentative schedule and reference sections in the book; lecture slides, assignments, help docs, …
- Book: *Database Systems: The Complete Book*, by H. Garcia-Molina, J. D. Ullman, and J. Widom. 2nd Ed.
- Programming: VM required, need significant programming on different platforms and languages
- Prerequisite: CompSci 201 - or you would have to learn some concepts yourself
- Q&A on Piazza
- Grades, sample solutions on Sakai
- Submissions on Gradescope and Gradiance
- Watch your email for announcements

Important: Grading

Absolute but adjustable grading

Guarantees:

[90%, 100%] A- / A / A+
[80%, 90%) B- / B / B+
[70%, 80%) C- / C / C+
[60%, 70%) D

Class topper gets A+

- Scale will not go upwards but can get downwards (e.g., based on the class performance in the exams)

- We will give you a feedback on your approximate standing after the midterm.

Duke Community Standard

- See course website for link
- Group discussion for assignments is okay (and encouraged), but
  - Acknowledge any help you receive from others
  - Make sure you "own" your solution
- All suspected cases of violation will be aggressively pursued

Course load

- (See course webpage for full details)

- Weekly (short) homework assignments (25%)
  - Each homework has same weight
  - Released on Tuesdays and due next Tuesday night (mostly)
  - Gradiance: immediately and automatically graded
  - Gradescope: programming problems, immediate feedback, later also manual grading
  - Gradescope: written solution, manual grading

- Midterm and final (20% each)
  - Open book, open notes
  - No communication/Internet whatsoever
  - Final is comprehensive, but emphasizes the second half of the course

Course load (contd.)

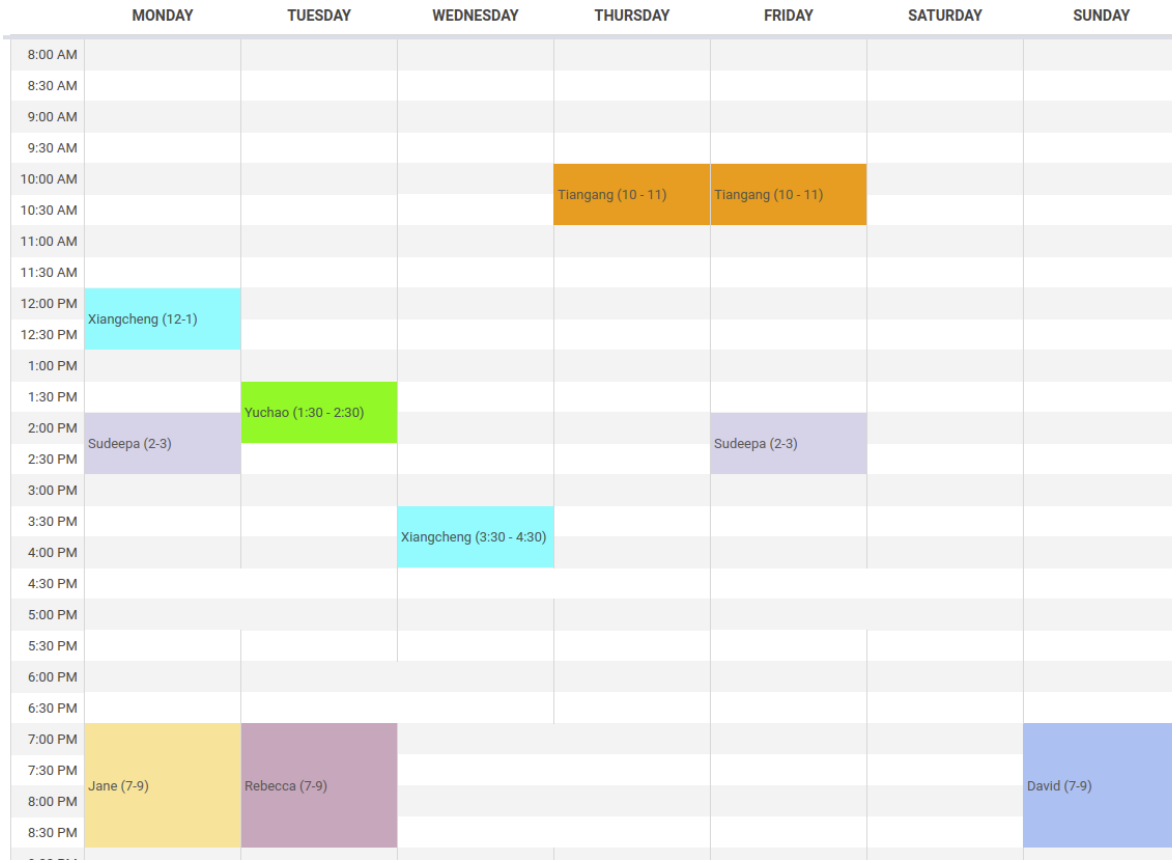
- Course project (20%)
  - Details to be given in the next 1-2 weeks

- In-class quiz (5%)
  - To review concepts right away in class – will be open for 5-10 mins
  - Will be announced at least one class in advance and on piazza
  - Each quiz: 50% for attempt on time and 50% for correct solution
  - Lowest score will be dropped (each quiz has same weight)

- In-class labs (5%)
  - Practice problems in class (both programming and conceptual) – each lab has the same weight
  - Will be announced at least one class in advance and on piazza
  - Due by the next day after class, 10% bonus points for finishing all problems in class correctly
  - TAs will be around to help you

Tentative office hours schedules

- Locations: TBD.
- See the updated info on the webpage
- More office hours around Tuesday (hws due), but good to start early!



Projects from past years

- RA: next-generation relational algebra interpreter
  - You may get to try it out for Homework #1!
- *Managing tent shifts and schedules!*
- *Tutor-tutee matching*
- *What's in my fridge and what can I cook?*
- *Hearsay: manage your own musics*
- *Dining at Duke (and deliver meals to students)*
- *National Parklopedia*: a website to find information about national parks

- *More examples later - but we expect you to be creative with a new idea!*

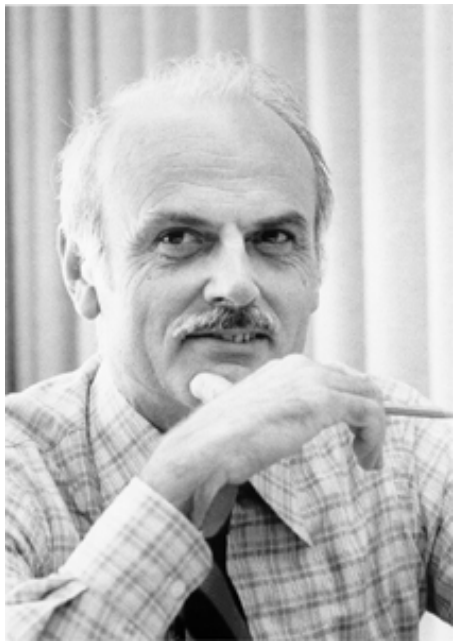
Let's get started!

Relational Data Model

What is a good model to store data?
Tree? Nested data? Graph?

(just) Tables!

Edgar F. Codd (1923-2003)



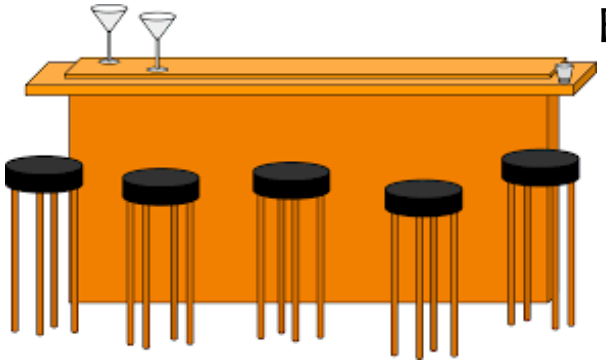
- Pilot in the Royal Air Force in WW2
- Inventor of the relational model and algebra while at IBM
- Turing Award, 1981

RDBMS = Relational DBMS

http://en.wikipedia.org/wiki/File:Edgar_F_Codd.jpg

The famous "Beers" database

Bars
Each has an address



Drinkers **Frequent** Bars
"X" times a week

Bars **Serve** Beers
At price "Y"

Drinkers **Likes** Beers

Drinkers
Each has an address

Beers
Each has a brewer

(Later in ER diagram – how to design a relational database)

"Beers" as a Relational Database

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Beer

Name	brewer
Budweiser	Anheuser-Busch Inc.
Corona	Grupo Modelo
Dixie	Dixie Brewing

Drinker

name	address
Amy	100 W. Main Street
Ben	101 W. Main Street
Dan	300 N. Duke Street

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Likes

drinker	beer
Amy	Corona
Dan	Budweiser
Dan	Corona
Ben	Budweiser

Relational data model

- A database is a collection of relations (or tables)
- Each relation has a set of attributes (or columns)
- Each attribute has a name and a domain (or type)
  - Set-valued attributes are not allowed
- Each relation contains a "set" of tuples (or rows)
  - Each tuple has a value for each attribute of the relation
  - Duplicate tuples are not allowed (Two tuples are duplicates if they agree on all attributes)
  - Ordering of rows doesn't matter (even though output is always in some order)
- However, SQL supports "bag" or duplicate tuples (why?)

☞ Simplicity is a virtue
  - not a weakness!

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

Schema vs. instance

- Schema
  - *Beer* (*name* string, *brewer* string)
  - *Serves* (*bar* string, *beer* string, price float)
  - *Frequents* (*drinker* string, bar string, times_a_week int)
- Instance
  - Actual tuples or records

☞Compare to types vs. collections of objects of these types in a programming language

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

Beer

Name	brewer
Budweiser	Anheuser-Busch Inc.
Corona	Grupo Modelo
Dixie	Dixie Brewing

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

SQL: Querying a RDBMS

- SQL: Structured Query Language
  - Pronounced "S-Q-L" or "sequel"
  - The standard query language supported by most DBMS
  - First developed at IBM System R
  - Follows ANSI standards

SQL is Declarative:

Programmer specifies what answers a query should return,
but not how the query is executed

DBMS picks the best execution strategy based on availability of indexes, data/workload characteristics, etc.
☞Provides physical data independence

Not a "Procedural" or "Operational" language like C++, Java, Python

Basic queries: SFW statement

- SELECT $A_1, A_2, \ldots, A_n$
  FROM $R_1, R_2, \ldots, R_m$
  WHERE $condition$

- SELECT, FROM, WHERE are often referred to as SELECT, FROM, WHERE "clauses"

Example: reading a table

- SELECT *
  FROM Serves

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

- Single-table query
- WHERE clause is optional
- * is a short hand for “all columns”

Example: selecting few rows

- SELECT beer AS mybeer
  FROM Serves
  WHERE price < 2.75

- SELECT beer
  FROM Serves
  WHERE bar = 'The Edge'

Serves

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

What does these return?

- SELECT list can contain expressions
  Can also use built-in functions such as SUBSTR, ABS, etc.
- String literals (case sensitive) are enclosed in single quotes
- "AS" is optional
- Do not want duplicates? Write SELECT DISTINCT beer …

Example: Join

- Find addresses of all bars that ‘Dan’ frequents

- Which tables do we need?

Example: Join

- Find addresses of all bars that 'Dan' frequents

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Beer

Name	brewer
Budweiser	Anheuser-Busch Inc.
Corona	Grupo Modelo
Dixie	Dixie Brewing

Drinker

name	address
Amy	100 W. Main Street
Ben	101 W. Main Street
Dan	300 N. Duke Street

bar	beer	price
The Edge	Budweiser	2.50
The Edge	Corona	3.00
Satisfaction	Budweiser	2.25

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Frequents

drinker	beer
Amy	Corona
Dan	Budweiser
Dan	Corona
Ben	Budweiser

Likes

Which tables do we need?

How do we combine them?

Example: Join

- Find addresses of all bars that 'Dan' frequents

```
SELECT B.address
FROM Bar B, Frequents F
WHERE B.name = F.bar
  AND F.drinker = 'Dan'
```

- Okay to omit *table_name* in *table_name.column_name* if *column_name* is unique
- Can use "Aliases" for convenience
  - "Bar as B" or "Bar B"

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Frequents

Semantics of SFW

- SELECT $E_1, E_2, \ldots, E_n$
  FROM $R_1, R_2, \ldots, R_m$
  WHERE $condition$

- For each t_1 in R_1:
  For each t_2 in R_2: … …
  For each t_m in R_m:

1. Apply "FROM"
Form cross-product of R1, .., Rm

If $condition$ is true over $t_1, t_2, \ldots, t_m$:

2. Apply "WHERE"
Only consider satisfying rows

Compute and output $E_1, E_2, \ldots, E_n$ as a row

3. Apply "SELECT"
Output the desired columns

Step 1: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" is outputs!

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Form Cross product of two relations

name	address	drinker	bar	times_a_week
The Edge	108 Morris Street	Ben	Satisfaction	2
The Edge	108 Morris Street	Dan	The Edge	1
The Edge	108 Morris Street	Dan	Satisfaction	2
Satisfaction	905 W. Main Street	Ben	Satisfaction	2
Satisfaction	905 W. Main Street	Dan	The Edge	1
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

Step 2: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" is outputs!

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Discard rows that do not satisfy WHERE condition

name	address	drinker	bar	times_a_week
~~The Edge~~	~~108 Morris Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
The Edge	108 Morris Street	Dan	The Edge	1
~~The Edge~~	~~108 Morris Street~~	~~Dan~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Dan~~	~~The Edge~~	~~1~~
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

Step 3: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" is outputs!

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

Output the "address" output of rows that survived

name	address	drinker	bar	times_a_week
~~The Edge~~	~~108 Morris Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
The Edge	108 Morris Street	Dan	The Edge	1
~~The Edge~~	~~108 Morris Street~~	~~Dan~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Ben~~	~~Satisfaction~~	~~2~~
~~Satisfaction~~	~~905 W. Main Street~~	~~Dan~~	~~The Edge~~	~~1~~
Satisfaction	905 W. Main Street	Dan	Satisfaction	2

Final output: Illustration of Semantics of SFW

- NOTE: This is "NOT HOW" the DBMS outputs the result, but "WHAT" is outputs!

Output the "address" output of rows that survived

- SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

Bar

name	address
The Edge	108 Morris Street
Satisfaction	905 W. Main Street

Frequents

drinker	bar	times_a_week
Ben	Satisfaction	2
Dan	The Edge	1
Dan	Satisfaction	2

address
108 Morris Street
905 W. Main Street

Try some SQL queries yourself on pgweb!

(See how to access the pgweb interface for a small "Beers" database on the slides posted on the course website)

Next: semantics of SFW statements in SQL

Announcements (Tue, 01/09)

- You should be on Sakai, Piazza, Gradescope
  - If you are not there or recently enrolled, please contact the instructor
- You will receive instructions on installing the VM
  - Please follow Piazza posts, all notifications will be posted there and you should receive emails right away
- First homework to be released on next class Tuesday 01/14, due in a week
  - No in-class quiz or labs unless explicitly announced in the class before (and posted on Piazza)