

Introduction

Introduction to Databases

CompSci 316 Fall 2022



DUKE
COMPUTER SCIENCE

Welcome to

CompSci 316: Introduction to Database Systems!!
Fall 2022

About us...

- Instructor: [Sudeepa Roy](#)
 - Associate Professor of Computer Science
 - At Duke CS since Fall 2015
 - PhD. UPenn, Postdoc: U. of Washington
 - Member of “Duke Database Devils” a.k.a. the database research group
- Research interests:
 - “data”
 - data management, database theory, data analysis, causality and explanations, data repair, query optimization...



Teaching Associate: [Alex Chao](#) (not ->)

Remember to copy Alex on the emails sent to Sudeepa!

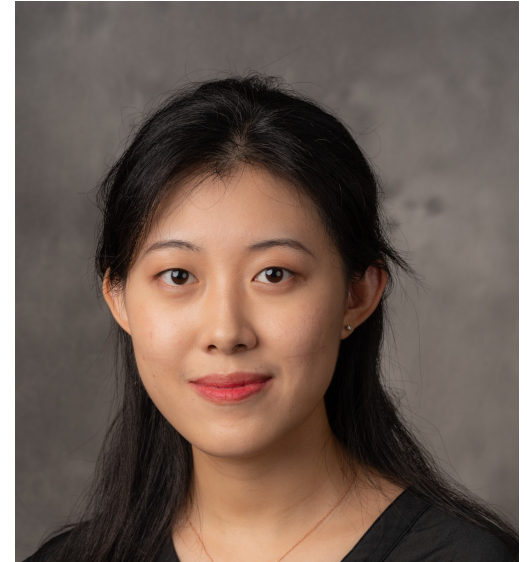
Only logistics questions specific to your situation should be sent to Sudeepa+Alex
everything else should be discussed on Ed



Yuxi Liu



Haibo Xiu

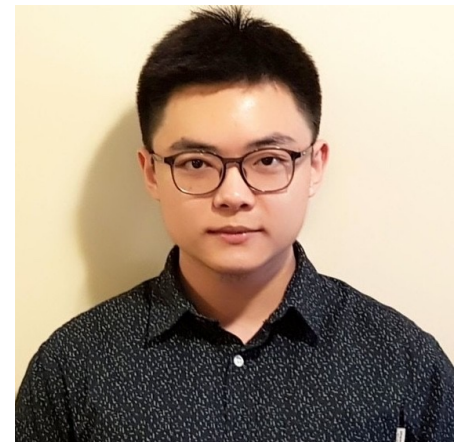


Tong Lin

Graduate TAs,
UTAs next slide!



Fiona Wu



Zhe Wang



Konstantinos
Bailas



Samy Boutouis



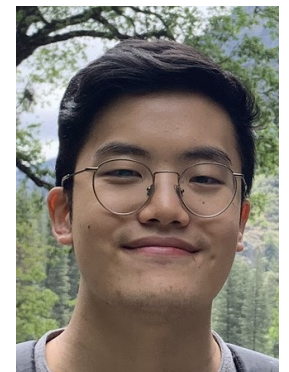
Neel Gajjar



Joshua Guo



Alexandra
Lawrence



Joon Young
Lee



Justin Lim



Danny Luo



Alok Malhotra



Harris Masterson



Jason Qiu



Alex Schiff



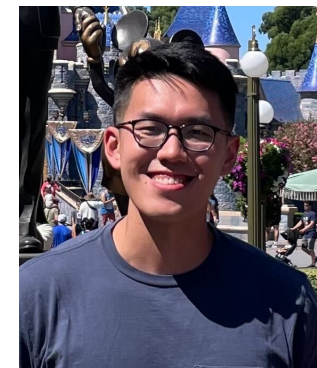
Grace Tian



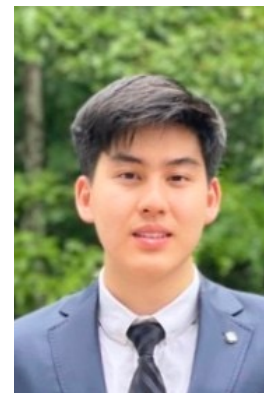
Joyce Wang



Samia Zaman



Han Zhang



Zachary
Zheng

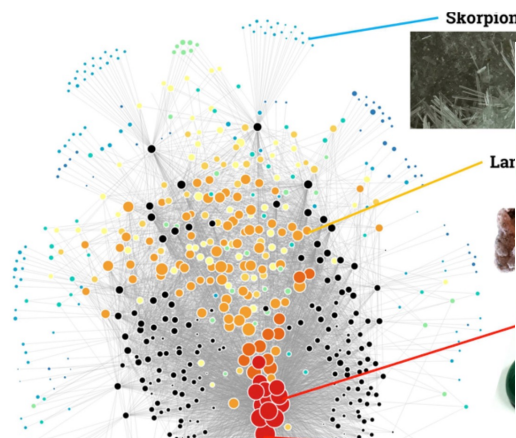
What are the goals of this course?

- 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 |



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

The New York Times

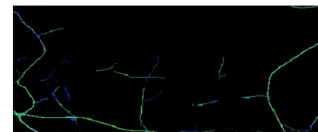
When Sports Betting Is Legal, the Value of Game Data Soars



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



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. Sailing 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 Statistics course for those things?

Yes, but..

Data doubles every 9 month – processing power only in 18 months..

Moore's Law vs. Parkinson's Law (disk sales in bits..)

Time to process all data doubles every 18 months!

Does your attention span or 24 hours per day double every 18 months?

No, so we need smarter data management and processing techniques!

1 TERABYTE A \$200 hard drive that holds 260,000 songs.	20 TERABYTE Photos uploaded to Facebook each month.	120 TERABYTE All the data and images collected by the Hubble Space Telescope.	330 TERABYTE Data that the large Hadron collider will produce each week.
460 TERABYTE All the digital weather data compiled by the national climate data center.	530 TERABYTE All the videos on Youtube.	600 TERABYTE ancestry.com's genealogy database (includes all U.S. census records 1790-2000)	1 PETABYTE Data processed by Google's servers every 72 minutes.

http://www.micronautomata.com/big_data



... So we 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 mini-Amazon* or a Book Selling Platform
- Large data! (think about all books in the world or even in English)

• How do we start?

* Many of you are going to do this in the course projects!

The class will be interactive...

Ask any question

Share any thoughts you have

Brainstorm ideas together

No question is too simple to ask!

Who are the key people?
(book-selling website)

Who are the key people? (book-selling website)

- 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

Sounds simple!

James Morgan#Durham, NC

... ..

A Tale of Two Cities#Charles Dickens#3.50#7

To Kill a Mockingbird#Harper Lee#7.20#1

Les Miserables#Victor Hugo#12.80#2

... ..

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

Query by programming

James Morgan#Durham, NC

... ..

A Tale of Two Cities#Charles Dickens#3.50#7

To Kill a Mockingbird#Harper Lee#7.20#1

Les Miserables#Victor Hugo#12.80#2

... ..

- 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?

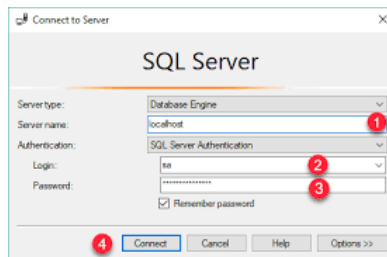
Revisit: What are the desired and necessary properties of the platform?

- Should be able to handle a **large amount of data**
 - Try to open a 10-100 GB file
- Should be **efficient** and **easy to use** (e.g., search with authors as well as title)
 - Try to search both on a large flat file
- 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
 - Imagine programmer's task
- 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
 - Imagine adding a new book or updating Copies (+ allow search) on a 10-100 GB text file
- Easy to **update and program**
 - For the admin

Solution?









- DBMS = Database Management System



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 power, information should not be lost or inconsistent 
- If 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 
 - Imagine one last copy of a book that two users purchase at the same time
- Easy to update and program 
 - For the admin

Optimization

Index

Recovery

Concurrency Control

Declarative

* We will learn these in the course!

DBMS helps the big ones!

The screenshot shows the MySQL website's 'Customers' page. The main heading is 'MySQL Customer: Facebook'. Below the heading is the Facebook logo and a quote: "We are one of the largest MySQL web sites in production. MySQL has been a revolution for young entrepreneurs." The page includes navigation links for Products, Cloud, Services, Partners, Customers, Why MySQL?, News & Events, and How to Buy. A sidebar on the left offers options like 'Customer Overview', 'Case Studies', and 'View By: Industry'.

The screenshot shows a Google Cloud documentation page titled 'Querying Cloud Bigtable data'. It explains how to use BigQuery to query data stored in Cloud Bigtable. The page includes a table of contents on the right with sections like 'Supported regions and zones', 'Retrieving the Cloud Bigtable URI', and 'Access controls and scopes'. A 'Beta' badge is visible at the bottom of the main content area.

The screenshot shows Mark Zuckerberg's Facebook profile. It features a profile picture, a cover photo, and a bio. The bio includes his role as Founder and CEO of Facebook, his education at Harvard University, and his residence in Palo Alto, California. A post from August 21 at 5:51 PM is visible, discussing election interference and the company's responsibility to protect users.

The screenshot shows the MySQL website's 'Customers' page with a grid of logos for various companies. The logos include GitHub, Facebook, and Italtel. Each logo is accompanied by a short testimonial and a 'Learn More' button. The page is categorized by industry, with 'Web: Ecommerce & Social' being the active category. Other categories include Technology: Hardware & Software, Telecom, Aerospace, Defense, Education, and Financial Services.

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, also for programmers for writing better queries)
 - Storage, index
 - Query processing, join algorithms, query optimizations
 - Transactions: recovery and concurrency control
- Glimpse of advanced topics and other DBMS
 - NOSQL, Spark (big data)
 - Data mining, Parallel DBMS
- Hands-on experience in class projects by building an end-to-end website or an app that runs on a database

Misc. course info

- All information available on the Course Website:
<https://www2.cs.duke.edu/courses/fall22/compsci316d/>
 - 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, info on Google credit to be updated
- Prerequisite: **CompSci 201 + CompSci 210/250**, or you would have to learn some concepts yourself
- Q&A on **Ed Discussion Board**
- Grades, sample solutions on **Sakai**
- Submissions on **Gradescope** and **Gradiance**
- Watch Ed for announcements
- Reach out to **both Sudeepa and Alex** only for questions on logistics applicable only to you

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

- Scale will not go upwards but can get downwards, i.e., grades go higher (e.g., if an exam is too hard)
 - If an exam is too easy – it is totally my responsibility
- You should have a fair idea about where you stand at any point in class

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
 - Cannot find solution from the Web and cannot get help anyone who is not in this class
- All suspected cases of violation will be aggressively pursued
- If you are unsure – ask on Ed or email Sudeepa/Alex

Course load / Grading

- (See course webpage for full details and late policy for each)
- Homework assignments (25%) + Gradiance (9%)
 - Each homework has same weight, due in about 7-10 days
 - Gradescope for programming problems, some with immediate feedback, some written solutions and manual grading
 - **Gradiance**: immediately and automatically graded, highest score recorded, no extension, lowest 2 scores dropped
- Midterm (17%) and final (18% each)
 - Open book, open notes
 - No communication/Internet whatsoever
 - Final is comprehensive, but may emphasize the second half of the course

Course load / Grading (contd.)

- **Course project (24%)**
 - Details to be given in the next 1-2 weeks
 - Strictly 5 members unless class size % 5 is not zero – then some groups with 4 or 6 members with permission from instructors
- **Discussion Sessions (5%)**
 - To practice problems, programming or set up help, project progress etc., with help from multiple TAs
 - Lowest three scores dropped
- **Communication (2%)**
 - For feedback and information asked from the students, all required
- **Extra credit (2% above 100%)**
 - Extra credit problems in HW and exams – lowest 25% scores dropped

Projects

- Fixed project Option: Mini-amazon
- Open project Option: Your own idea! (More work, more fun)
 - From previous 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*
- *Browse through Lec-1 of Fall'21 for more examples*
- *Project-details doc will be posted in the next 1-2 weeks*
- *Start looking for 4 other members from the same discussion session – think whether you want to do the fixed project with more specifications provided or be creative in an open project!*

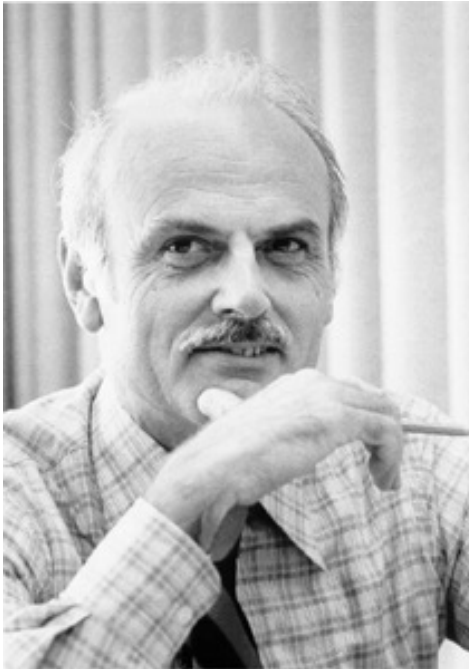
Let's get started!

Relational Data Model

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

(just) **Tables!**

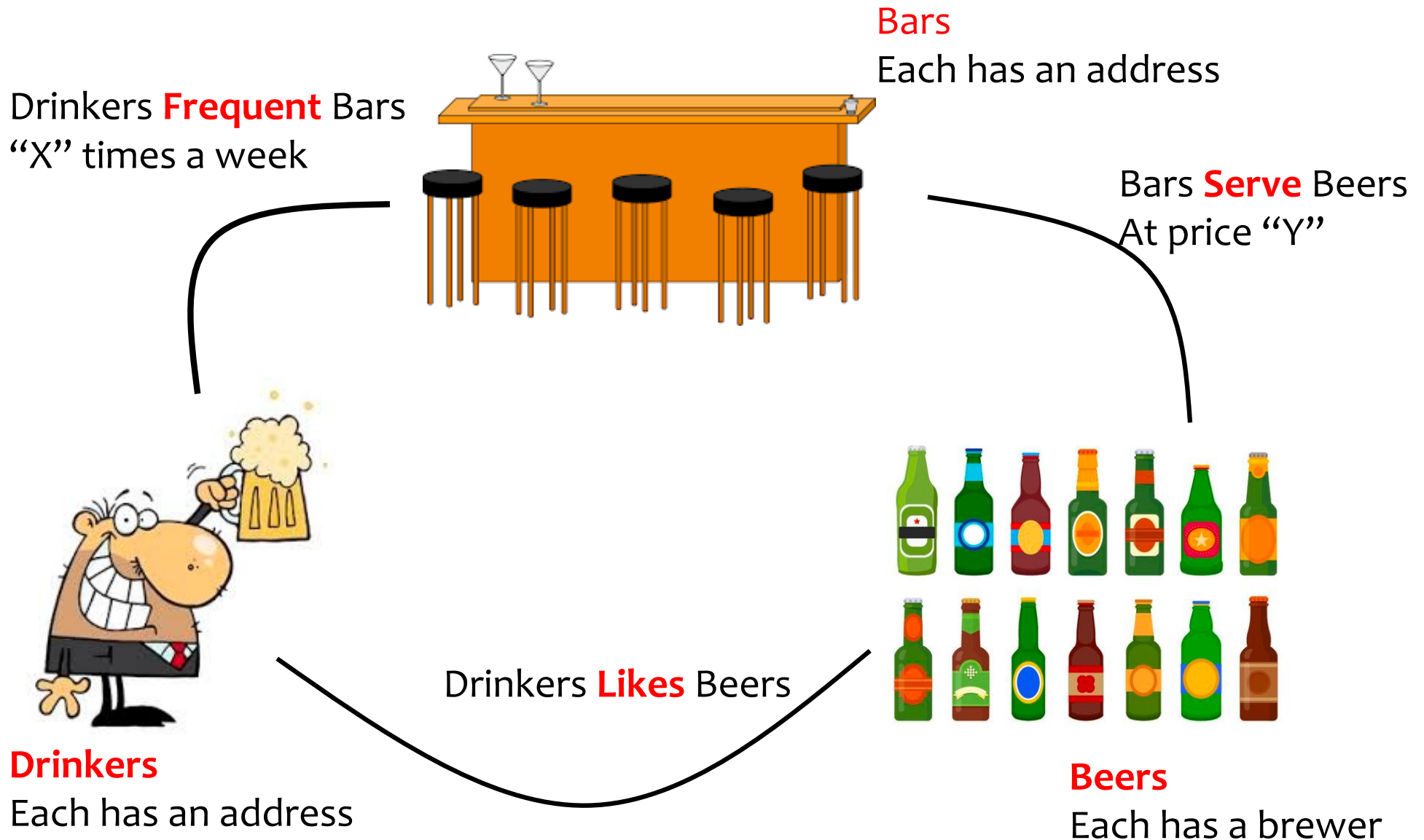
Edgar F. Codd (1923-2003)



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

RDBMS = Relational DBMS

The famous “Beers” database



(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

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

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 (e.g., you cannot store a list/set of bars in a cell, all cells have to contain atomic values)
- 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

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

Serves

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

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

Beer

Frequents

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

Announcements (Tue, 08/30)

- You are/will be on Sakai, Ed, Gradescope by the next class
- First discussion session on Friday about setting up VM and some practice problems
- Please follow Ed posts, all notifications will be posted there
- First homework to be released soon

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, \dots, A_n
FROM R_1, R_2, \dots, 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`

Serves

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

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

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'
```

**Bar**

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

- 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”

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

**Frequents**