Introduction

Introduction to Databases
CompSci 316 Fall 2021
About us: instructor and TA

- Instructor: Jun Yang
  - Started at Duke in 2001
  - Been doing (and enjoying) research in databases ever since grad school (1995)
    - Didn’t take any database as an undergrad 😱
  - Now working on data-intensive systems and computational journalism

- Graduate TAs:
  - Yihao Hu
  - Rich Stureborg
  - PhD students in Computer Science

- UTAs: still forming the team
What comes to your mind...

...when you think about “databases”?
But these use databases too…

Facebook used MySQL to store posts, for example (at least as of 2017)

WordPress uses MySQL to manage components of a website (pages, links, menus, etc.)

Introduction to Database Systems (Fall 2021)
COMPSCI 316-01, Duke University

Home

When: TTh 3:15PM – 4:00PM
Where: Bryan Center Griffith Theater
Instructor: Jian Yang
Grad TAs: Yihao Hu and Rich Zurek
UTAs: TBD

We intend this course to give you a solid background in database systems as well as managing and processing "big data" in general. Topics include data modeling, database design theory, data definition and manipulation languages (SQL and NoSQL), database application programming interfaces, storage and indexing, query processing and optimization, parallel and distributed data processing, transaction processing, as well as a sample of other topics such as data mining and web data. Programming projects are required.

Prerequisites: CompSci 201 or equivalent, or consent of the instructor. You will need familiarity (or ability to quickly become familiar) with the Unix command line (such as "Terminal" in Mac OS).

Special thanks to Google for their support of Google Cloud credits for this course!
The final system was able to collect and store 5 petabytes of data. If 1 byte were a 2-foot-by-2-foot tile, then 1 petabyte would cover the whole Earth. But the job of converting that data into an image required the creation of entirely new software tools. https://coronavirus.jhu.edu/map.html (retrieved 2021-08-22)
These tools reveal where our forecasts need some tweaking. But we also think they show that FiveThirtyEight’s models have performed strongly. All of our forecasts have proved to be more valuable than an unskilled guess, and things we say will happen only rarely … tend to happen only rarely.

March Madness (men)
Data → power

**Census data sets up redistricting fight over growing suburbs**

By DAVID A. LIEB and ACACIA CORONADO  August 13, 2021

https://apnews.com/article/census-2020-house-elections-redistricting-cbad52d72fadac3a5688d6a0036818f4

**BROOKINGS**

**The disturbing implications of increasingly narrow political ad targeting**

February 11, 2021 | Katie Joseff, Joel Carter, and Samuel Woolley

... political operatives are using such data to carry out increasingly granular geo-targeted advertising—sometimes so granular that it is used to target one individual. But these operatives are finding that demographic data allows for more invasive targeting than location data and are beginning to target messages at niche audiences through the coordination of small-scale social media influencers...

https://www.brookings.edu/techstream/the-disturbing-implications-of-increasingly-narrow-political-ad-targeting/

**THE WALL STREET JOURNAL.**

**Facebook Disables Access for NYU Research Into Political-Ad Targeting**

Researchers say the data provided information on misinformation in political ads; Facebook says the work violated its terms of service

By Meghan Bobrowsky

Updated Aug. 4, 2021 5:54 pm ET

https://www.wsj.com/articles/facebook-cuts-off-access-for-nyu-research-into-political-ad-targeting-11628052204
Democratizing data (and analysis)

- Democratization of data: more data—relevant to you and the society—are being collected
  - “Smart planet”: sensors for phones and cars, roads and bridges, buildings and forests, ...
  - “Government in the sunshine”: spending reports, school performance, crime reports, corporate filings, campaign contributions, ...
- But few people know how to analyze them
  - Even fewer know how to analyze them responsibly
- You will learn how to help bridge this divide
Computational challenge

• Moore’s Law:  
  *Processing power doubles every 18 months*

• But *amount of data doubles every 9 months*
  • Disk sales (# of bits) doubles every 9 months
  • Parkinson’s Law: 
    *Data expands to fill the space available for storage*

<table>
<thead>
<tr>
<th></th>
<th>1 TERABYTE</th>
<th>20 TERABYTE</th>
<th>120 TERABYTE</th>
<th>330 TERABYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A $200 hard drive that holds 260,000 songs.</td>
<td>Photos uploaded to Facebook each month.</td>
<td>All the data and images collected by the Hubble Space Telescope.</td>
<td>Data that the large Hadron collider will produce each week.</td>
</tr>
<tr>
<td>460 TERABYTE</td>
<td>All the digital weather data compiled by the national climate data center.</td>
<td>530 TERABYTE</td>
<td>All the videos on Youtube.</td>
<td>600 TERABYTE</td>
</tr>
<tr>
<td>1 PETABYTE</td>
<td>1 PETABYTE</td>
<td>Data processed by Google’s servers every 72 minutes.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://www.micronautomata.com/big_data
Moore’s Law reversed

Time to process all data doubles every 18 months!

• Does your attention span double every 18 months?
  • No, so we need smarter data management and processing techniques
Misc. course info

• Website: https://sites.duke.edu/compsci316_01_f2021/
  • 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. 2\textsuperscript{nd} Ed.

• Programming: VM required; $50 worth of credits for VMs in the cloud, courtesy of Google

• Q&A on Ed

• Grades, sample solutions on Sakai

• Watch your email for announcements

• Office hours to be posted
Grading

\[
\begin{align*}
[90\%, \infty) & \quad A- / A / A+ \\
[80\%, 90\%) & \quad B- / B / B+ \\
[70\%, 80\%) & \quad C- / C / C+ \\
[60\%, 70\%) & \quad D \\
[0\%, 60\%) & \quad F
\end{align*}
\]

• No “curves”

• Scores may be adjusted upwards if, for example, an exam is too difficult
  • Scores won’t go down—mistake would be mine alone if I made an exam too easy
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

• ~A dozen Gradiance exercises (10%)
  • Immediately and automatically graded

• Four homework assignments (25%)
  • Written and programming problems; submit through Gradescope

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

• Course project (25%)
  • Details to be given in the third week of class
Projects from recent years

- Duke Shares Lunch
  - Help prevent food points going to waste when year ends
  - Paul Dellinger, AJ Eckmann, Josh Romine, 2019

- BYEStander: Sexual Assault Prevention By Removing the Bystander Effect
  - Bystander effect: unwillingness to act first
  - A mobile platform to allow real-time anonymous reporting of suspicious activity and alerting all nearby users
  - Soomin Cho, Helena Merk, Ian Roughen, Katie Wilbur, Blaire Zhang, 2018

- Duke Fashion Auction Online
  - A full-fledged online auction system, inspired by the “Duke Fashion Exchange” Facebook page
  - Chunxi Ding, McCourt Hu, Amy Xiong, Ben Yang, Lin Zuo, 2018
Projects from earlier years

- **Duke Conversations**: helping to manage the program that hosts informal dinners with faculty and students to foster engagement on campus
  - Noah Burrell, Anne Driscoll, Kimberly Eddleman, Summer Smith, Sarp Uner, 2017

- **LegiToken**: help users research on ICOs (Initial Coin Offerings) by consolidating information from multiple sources and social media
  - Stuart Baker, Austin Carter, Alex Gerrese, Oscar Hong, Trent Large, Joshua Young, 2017

- **Partners for Success Tutoring App**: connecting volunteer tutors to Durham teachers and students
  - Justin Bergkamp, Cosette Goldstein, Sophie Polson, Bailey Wall, 2016

- **Congress Talking Points**: analyses (sentiment, similarity, etc.) of speeches

- **SMSmart** *(4.1 stars on Google Play)*: search/tweet/Yelp without data
  - Alan Ni, Jay Wang, Ben Schwab, 2014

- **FarmShots**: help farmers with analysis of satellite images
  - Acquired by Syngenta in Feb. 2018
More past examples

- **Pickup Coordinator**: app for coordinating carpool/pickups
  - Adam Cue, Kevin Esoda, Kate Yang, 2012
- **Mobile Pay**
  - Michael Deng, Kevin Gao, Derek Zhou, 2012
- **FriendsTracker app**: where are my friends?
  - Anthony Lin, Jimmy Mu, Austin Benesh, Nic Dinkins, 2011
- **ePrint iPhone app**
  - Ben Getson and Lucas Best, 2009
- **Making iTunes social**
  - Nick Patrick, 2006; Peter Williams and Nikhil Arun, 2009
- **Duke Schedulator**: ditch ACES—plan visually!
  - Alex Beutel, 2008
- **SensorDB**: manage/analyze sensor data from forest
- **Facebook**
  - Tyler Brock and Beth Trushkowsky, 2005
- **K-ville tenting management**
  - Zach Marshall, 2005
Your turn to be creative

http://www.yummymummyclub.ca/sites/default/files/styles/large/public/field/image/teaching_kidscreative_skills.jpg
So, what is a database system?

From Oxford Dictionary:

- **Database**: an organized body of related information
- **Database system, DataBase Management System (DBMS)**: a software system that facilitates the creation and maintenance and use of an electronic database
What do you want from a DBMS?

• Keep data around (**persistent**)
• Answer questions (**queries**) about data
• **Update** data

• Example: a traditional banking application
  • **Data**: Each account belongs to a branch, has a number, an owner, a balance, ...; each branch has a location, a manager, ...
  • **Persistency**: Balance can’t disappear after a power outage
  • **Query**: What’s the balance in Homer Simpson’s account? What’s the difference in average balance between Springfield and Capitol City accounts?
  • **Modification**: Homer withdraws $100; charge accounts with lower than $500 balance a $5 fee
Sounds simple!

- Text files
- Accounts/branches separated by newlines
- Fields separated by #’s

```
1001#Springfield#Mr. Morgan

... ...
00987-00654#Ned Flanders#2500.00
00123-00456#Homer Simpson#400.00
00142-00857#Montgomery Burns#1000000000.00
... ...
```
Query by programming

What’s the balance in Homer Simpson’s account?

A simple script

- Scan through the accounts file
- Look for the line containing “Homer Simpson”
- Print out the balance
Query processing tricks

• Tens of thousands of accounts are not Homer’s
  ❆ Cluster accounts by owner’s initial: those owned by “A...” go into file A; those owned by “B...” go into file B; etc. → decide which file to search using the initial
  ❆ Keep accounts sorted by owner name → binary search?
  ❆ Hash accounts using owner name → compute file offset directly
  ❆ Index accounts by owner name: index entries have the form \( \langle \text{owner\_name}, \text{file\_offset} \rangle \) → search index to get file offset
  ❆ And the list goes on...

What happens when the query changes to: What’s the balance in account 00142-00857?
Observations

• There are many techniques—not only in storage and query processing, but also in concurrency control, recovery, etc.

• These techniques get used over and over again in different applications

• Different techniques may work better in different usage scenarios
The birth of DBMS – 1

From Hans-J. Schek’s VLDB 2000 slides
The birth of DBMS – 2

From Hans-J. Schek’s VLDB 2000 slides
The birth of DBMS – 3

From Hans-J. Schek’s VLDB 2000 slides
Early efforts

• “Factoring out” data management functionalities from applications and standardizing these functionalities is an important first step
  • CODASYL standard (circa 1960’s)
    Bachman got a Turing award for this in 1973

• But getting the abstraction right (the API between applications and the DBMS) is still tricky
CODASYL

• Query: Who have accounts with 0 balance managed by a branch in Springfield?

• Pseudo-code of a CODASYL application:

  Use index on account(balance) to get accounts with 0 balance;
  For each account record:
    Get the branch id of this account;
    Use index on branch(id) to get the branch record;
    If the branch record’s location field reads “Springfield”:
      Output the owner field of the account record.

• Programmer controls “navigation”: accounts → branches
  • How about branches → accounts?
What’s wrong?

• The best navigation strategy & the best way of organizing the data depend on data/workload characteristics

With the CODASYL approach

• To write correct code, programmers need to know how data is organized physically (e.g., which indexes exist)

• To write efficient code, programmers also need to worry about data/workload characteristics

☞ Can’t cope with changes in data/workload characteristics
The relational revolution (1970’s)

• A simple model: data is stored in relations (tables)
• A declarative query language: SQL

```
SELECT Account.owner
FROM Account, Branch
WHERE Account.balance = 0
AND Branch.location = 'Springfield'
AND Account.branch_id = Branch.branch_id;
```

• 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
Physical data independence

• Applications should NOT worry about how data is physically structured and stored
• Applications should work with a **logical** data model and **declarative** query language
• Leave the implementation details and optimization to DBMS
• The single **most important reason** behind the success of DBMS today
  • And a Turing Award for E. F. Codd in 1981
Standard DBMS features

• Persistent storage of data

• Logical data model; declarative queries and updates → physical data independence
  • Relational model is the dominating technology today

☞ What else?
DBMS is multi-user

• Example
  get account balance from database;
  if balance > amount of withdrawal then
    balance = balance - amount of withdrawal;
    dispense cash;
    store new balance into database;

• Homer at ATM1 withdraws $100
• Marge at ATM2 withdraws $50
• Initial balance = $400, final balance = ?
  • Should be $250 no matter who goes first
Final balance = $300

Homer withdraws $100:
read balance; $400
if balance > amount then
    balance = balance - amount; $300
write balance; $300

Marge withdraws $50:
read balance; $400
if balance > amount then
    balance = balance - amount; $350
write balance; $350
Final balance = $350

Homer withdraws $100:
read balance; $400
if balance > amount then
  balance = balance - amount; $300
write balance; $300

Marge withdraws $50:
read balance; $400
if balance > amount then
  balance = balance - amount; $350
write balance; $350
Concurrency control in DBMS

• Similar to concurrent programming problems?
  • But data not main-memory variables

• Similar to file system concurrent access?
  • Lock the whole table before access
    • Approach taken by MySQL in the old days
    • Still used by SQLite (as of Version 3)
  • But want to control at much finer granularity
    • Or else one withdrawal would lock up all accounts!
Recovery in DBMS

• Example: balance transfer
decrement the balance of account X by $100;
increment the balance of account Y by $100;

• Scenario 1: Power goes out after the first operation

• Scenario 2: DBMS buffers and updates data in memory (for efficiency); before they are written back to disk, power goes out

• How can DBMS deal with these failures?
Standard DBMS features: summary

- Persistent storage of data
- Logical data model; declarative queries and updates → physical data independence
- Multi-user concurrent access
- Safety from system failures
- Performance, performance, performance
  - Massive amounts of data (terabytes~petabytes)
  - High throughput (thousands~millions transactions/hour)
  - High availability (≥ 99.999% uptime)
Standard DBMS architecture

- Much of the OS may be bypassed for performance and safety
- We will be filling in many details of the DBMS box throughout the semester
AYBABTU?

“Us” = relational databases

• Most data are not in them!
  • Personal data, web, scientific data, system data, ...

• Text and semi-structured data management
  • XML, JSON, ...

• “NoSQL” and “NewSQL” movement
  • MongoDB, Cassandra, BigTable, HBase, Spanner, HANA, Spark...

• This course will look beyond relational databases
Course components

• Relational databases
  • Relational algebra, database design, SQL, app programming

• Semi-structured data
  • Data model & query languages, app programming, interplay with relational databases

• Database internals
  • Storage, indexing, query processing and optimization, concurrency control and recovery

• Advanced topics (TBD)
  • Parallel data processing/MapReduce/Spark, data warehousing & data mining, Web search & indexing, etc.
Announcements (Tue. Aug. 24)

• Office hours for me and grad TAs start this week
  • Mine is right after lectures—today we will talk and walk towards North 232

• Need a spot in the class?
  • Given our current staff size, it’s hard to increase enrollment
  • Keep in mind that 316 will be offered in Fall 2022, and more advanced students can take 516 in Spring 2022

• Thursday: we will do relational algebra—the first of many query languages we shall learn