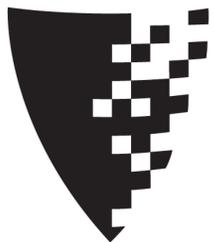


Introduction

Introduction to Databases

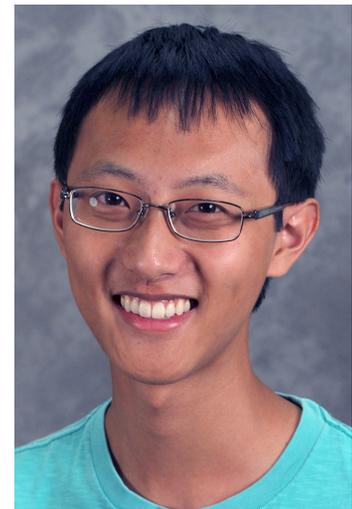
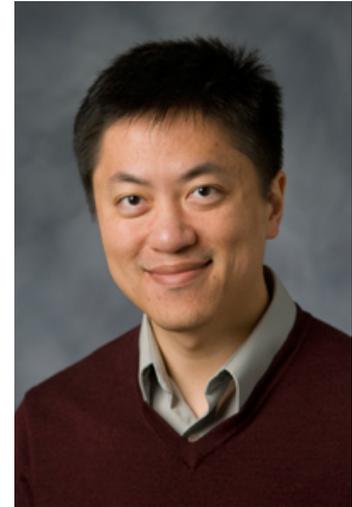
CompSci 316 Fall 2017



DUKE
COMPUTER SCIENCE

About us: instructor and TA

- Instructor: **Jun Yang**
 - 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 TA: **Andrew Lee**
 - PhD student in Computer Science
 - Working on “big data” analysis



About us: UTAs



Seon



Sophie



Amy



David

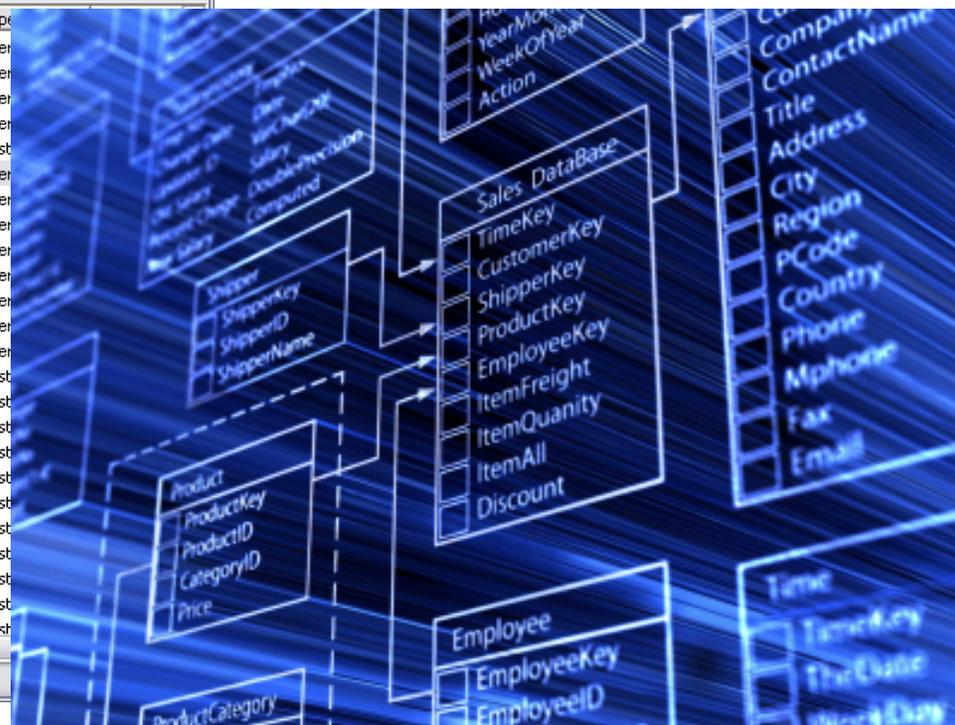
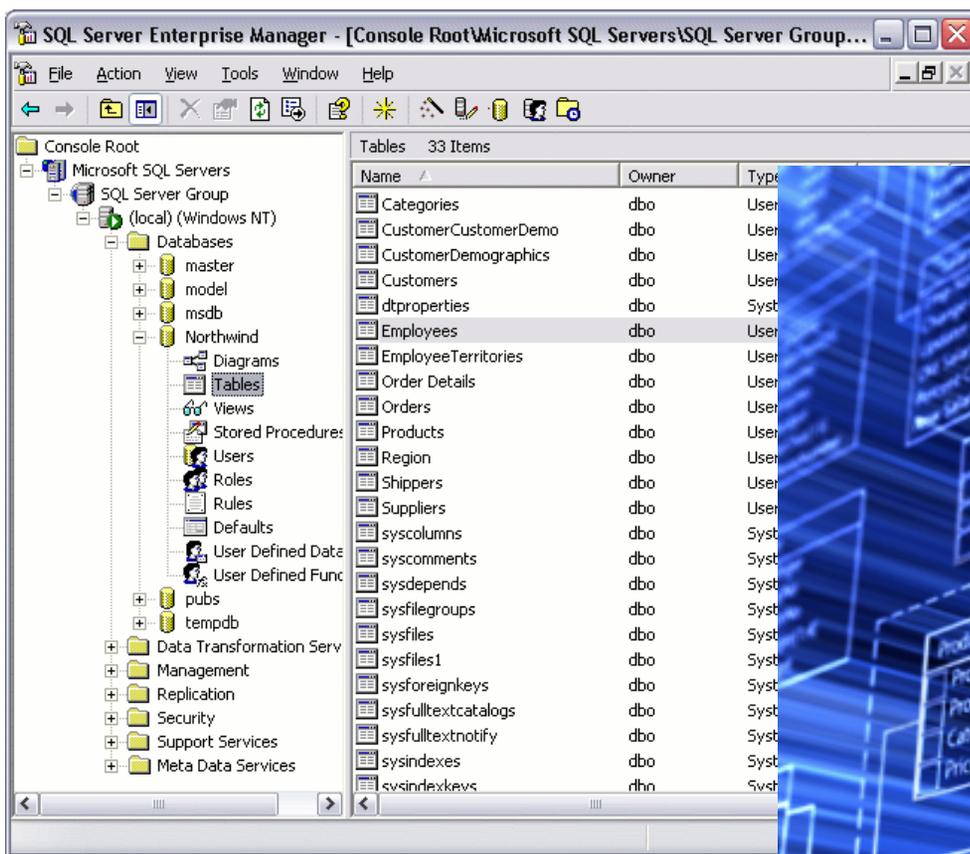


Jia

All CompSci 316 veterans!

What comes to your mind...

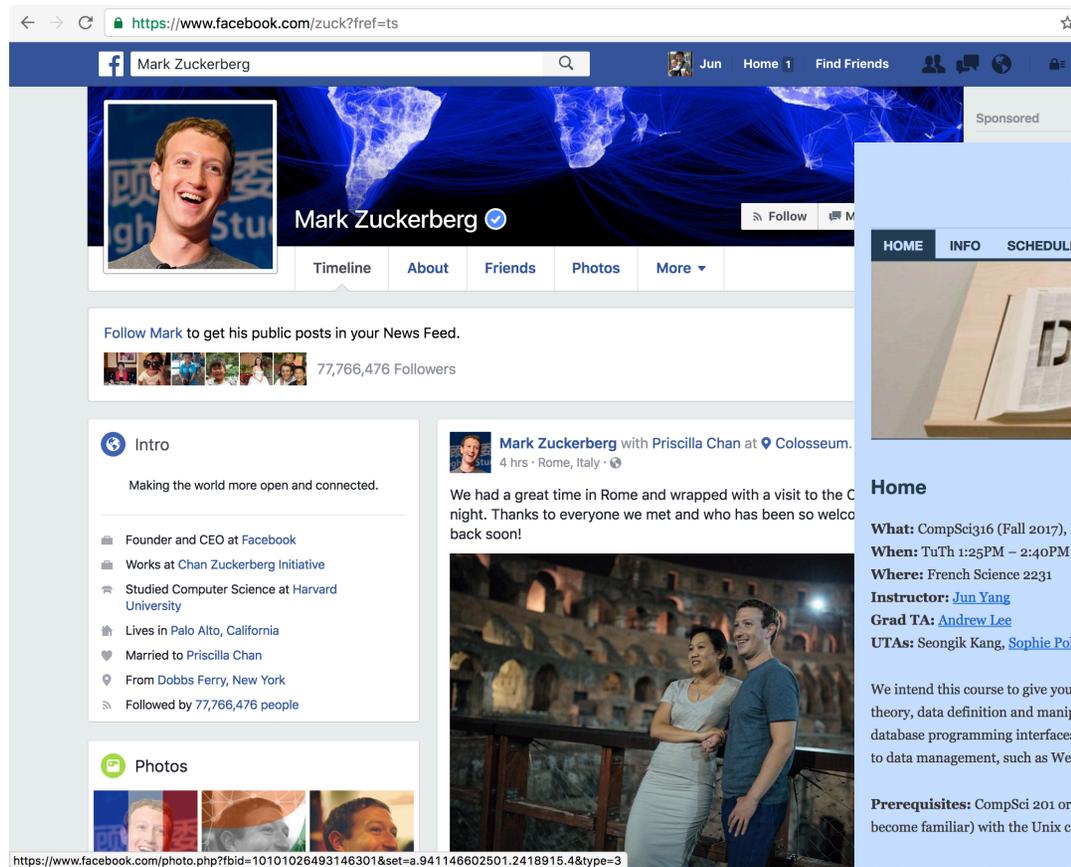
... when you think about “databases”?



http://www.quackit.com/pix/database/tutorial/dbms_sql_server.gif

<http://webstoresltd.com/wp-content/uploads/2013/06/database-design.jpg>

But these use databases too...



Facebook uses MySQL to store posts, for example

A screenshot of the course website for "Introduction to Databases (Fall 2017)" at Duke University. The page title is "Introduction to Databases (Fall 2017)" with the course ID "COMPSCI 316-01, Duke University". The navigation menu includes "HOME", "INFO", "SCHEDULE", "HELP", "GRADUANCE", "DISCUSSION (PIAZZA)", and "SUBMISSION & GRADES (SAKAI)". The main image shows two open books with the words "DATA" and "BASE" on the pages. The "Home" section contains the following information:

- What:** CompSci316 (Fall 2017), Duke University
- When:** TuTh 1:25PM – 2:40PM
- Where:** French Science 2231
- Instructor:** [Jun Yang](#)
- Grad TA:** [Andrew Lee](#)
- UTAs:** Seongik Kang, [Sophie Polson](#), [Amy Yang](#), [David Yang](#), [Jia Zeng](#)

The course description states: "We intend this course to give you a solid background in database systems as well as data management in general. Topics include data modeling, database design theory, data definition and manipulation languages, storage and indexing techniques, query processing and optimization, concurrency control and recovery, and database programming interfaces. Besides relational and semi-structured (e.g., XML and JSON) data, this course also samples a number of other topics related to data management, such as Web search, data warehousing, data mining, and "big data." Programming projects are required."

Prerequisites: CompSci 201 or equivalent, or consent of the instructor. At the minimum, you will need CompSci 101, and 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!

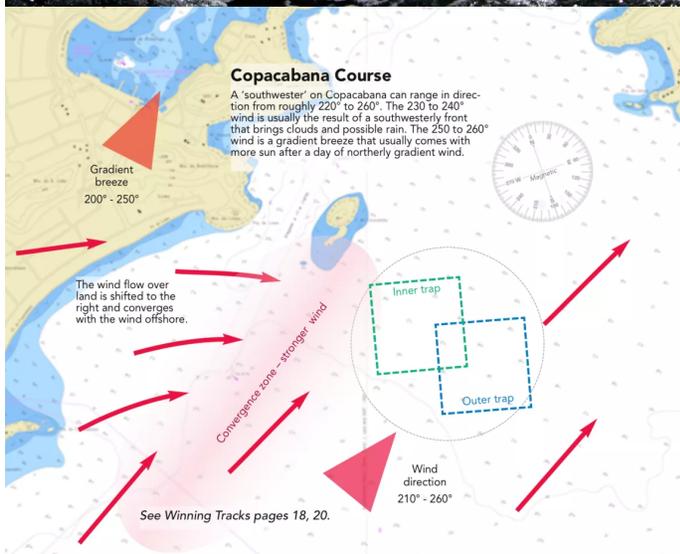
The footer of the page reads: "Proudly powered by WordPress. Theme: Coraline by WordPress.com."

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

Data → gold



... 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?”

Data → profit and fun



TECH | CONSUMER TECHNOLOGY

‘Pokémon Go’ Creator Closes Privacy Hole But Still Collects User Data

Players with iPhones should log out and download the update; game’s developers say it never snooped



Pokemon Go is displayed on a cell phone. PHOTO: ASSOCIATED PRESS



WHAT HAVE YOU FOUND, TRAVELER?



SIGHTING REPORT »

A NEST (MULTIPLE) »

SIGHTING REPORTS

are normal spawn reports - i.e. when you encounter a Pokémon in your travels, submit a Sighting Report!

NESTS

are our term for repeat spawns or clustered spawns of the same species. Only report a 'Nest' if you see multiples of an uncommon species at once (or a consistently spawning rare 'mon)!

... Silph’s newest initiative is to have travelers log the location of “nests,” spots where a certain species of monster is guaranteed to appear, and sometimes several instances of that species (e.g. Charmanders gather at New York’s Museum of Natural History.)

— GIZMODO

Data → power



The Data That Turned the World Upside Down

HG HANNES GRASSEGER & MIKAEL KROGERUS
Jan 28 2017, 9:15am

Psychologist Michal Kosinski developed a method to analyze people in minute detail based on their Facebook activity. Did a similar tool help propel

... a then little-known British company based in London sent out a press release: “We are thrilled that our revolutionary approach to data-driven communication has played such an integral part in President-elect Trump’s extraordinary win,” Alexander James Ashburner Nix was quoted as saying. Nix is British, 41 years old, and CEO of Cambridge Analytica. ... His company wasn’t just integral to Trump’s online campaign, but to the UK’s Brexit campaign as well.

Challenges

- 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

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			

Moore's Law reversed

*Time to process all data
_____ every 18 months!*

- Does your attention span double every 18 months?
 - No, so we need smarter data management and processing techniques

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**
- You will learn how to help bridge this divide

Misc. course info

- Website: http://sites.duke.edu/compsci316_01_f2017/
 - 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; \$50 worth of credits for VMs in the cloud, courtesy of Google
- Q&A on Piazza
- Submissions, grades, sample solutions on Sakai
- Watch your email for announcements
- Office hours to be posted

Grading

[90%, 100%]	A- / A / A+
[80%, 90%)	B- / B / B+
[70%, 80%)	C- / C / C+
[60%, 70%)	D
[0%, 60%)	F

- No “curves”
- Scale may be adjusted downwards (i.e., grades upwards) if, for example, an exam is too difficult
- Scale will not go upwards—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

- Four homework assignments (35%)
 - **Gradiance**: immediately and automatically graded
 - Plus written and programming problems
- Course project (25%)
 - Details to be given in the third week of class
- Midterm and final (20% each)
 - Open book, open notes
 - No communication/Internet whatsoever
 - Final is comprehensive, but emphasizes the second half of the course

Projects from last year

- *Partners for Success Tutoring App*: connecting volunteer tutors to Durham teachers and students
 - Justin Bergkamp, Cosette Goldstein, Sophie Polson (your UTA), Bailey Wall
- *Hunt*: iPhone app for team Scavenger hunt
 - Cody Li, Brian Lin, Andy Wang, David Yang (your UTA)
- *Congress Talking Points*: analyses (sentiment, similarity, etc.) of speeches by members of the Congress
 - Gautam Hathi, Joy Patel, Seon Kang (your UTA), Zoey Zou

Projects from earlier years

- **wikiblocks** (vimeo.com/147680387): find visualizations for Wiki pages
 - Brooks Mershon, Mark Botros, Manoj Kanagaraj, Davis Treybig, 2015
- **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
 - Ouwen Huang, Arun Karottu, Yu Zhou Lee, Billy Wan, 2014
- **FoodPointsMaster**: tracks balance & spending habit
 - Howard Chung, Wenjun Mao, William Shelburne, 2014
- **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

More past examples

- **ePrint iPhone app**
 - Ben Getson and Lucas Best, 2009
- **Making iTunes social**
 - Nick Patrick, 2006; Peter Williams and Nikhil Arun, 2009
- **Duke Scheduler: ditch ACES—plan visually!**
 - Alex Beutel, 2008
- **SensorDB: manage/analyze sensor data from forest**
 - Ashley DeMass, Jonathan Jou, Jonathan Odom, 2007
- **Facebook+**
 - Tyler Brock and Beth Trushkowsky, 2005
- **K-ville tenting management**
 - Zach Marshall, 2005



Your turn to be creative

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!

```
1001#Springfield#Mr. Morgan
```

```
... ..
```

```
00987-00654#Ned Flanders#2500.00
```

```
00123-00456#Homer Simpson#400.00
```

```
00142-00857#Montgomery Burns#1000000000.00
```

```
... ..
```

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

Query by programming

```
1001#Springfield#Mr. Morgan
```

```
... ..
```

```
00987-00654#Ned Flanders#2500.00
```

```
00123-00456#Homer Simpson#400.00
```

```
00142-00857#Montgomery Burns#1000000000.00
```

```
... ..
```

- 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

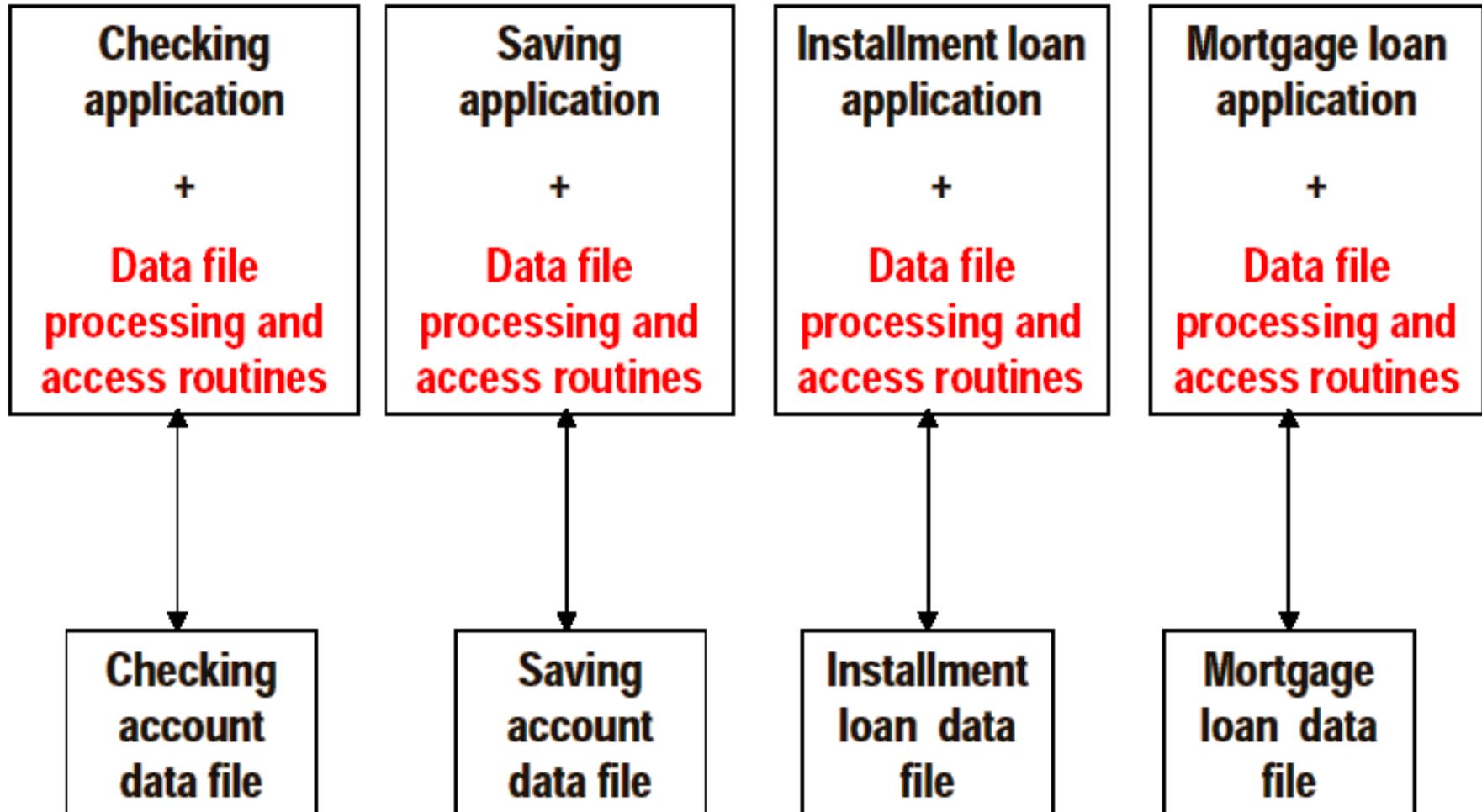
☞ And the list goes on...

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

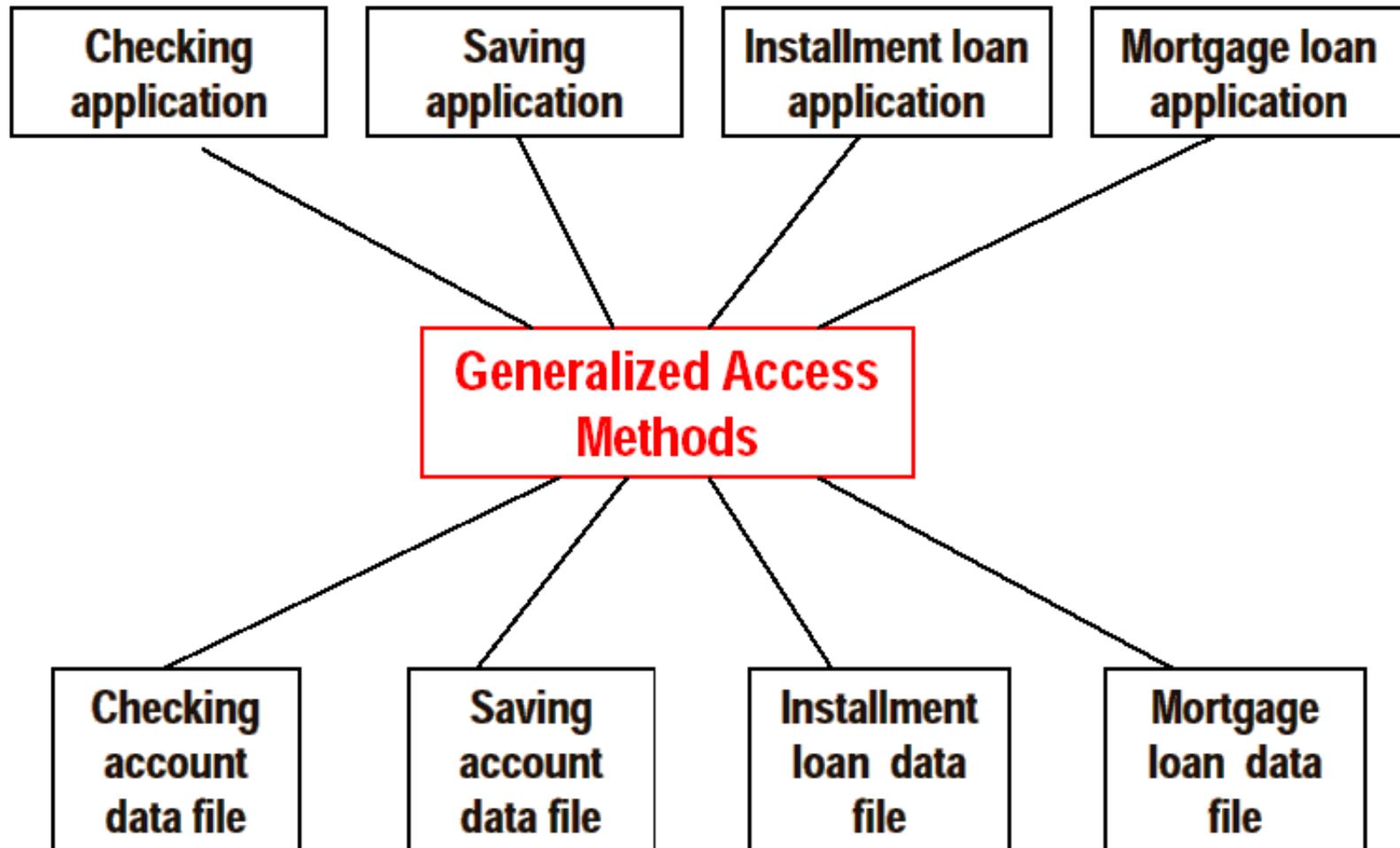
Observations

- 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

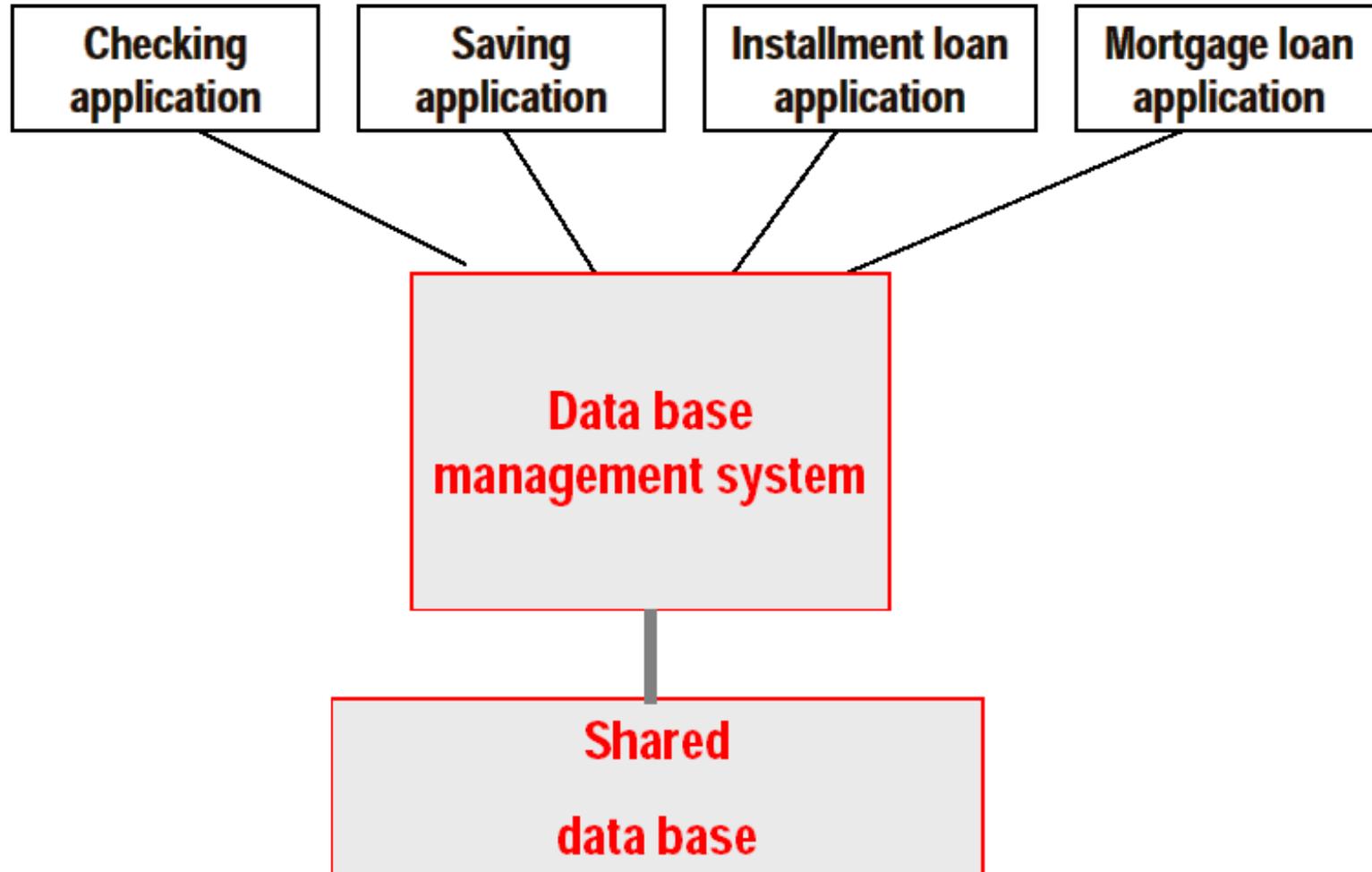
The birth of DBMS – 1



The birth of DBMS – 2



The birth of DBMS – 3



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 need to 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!

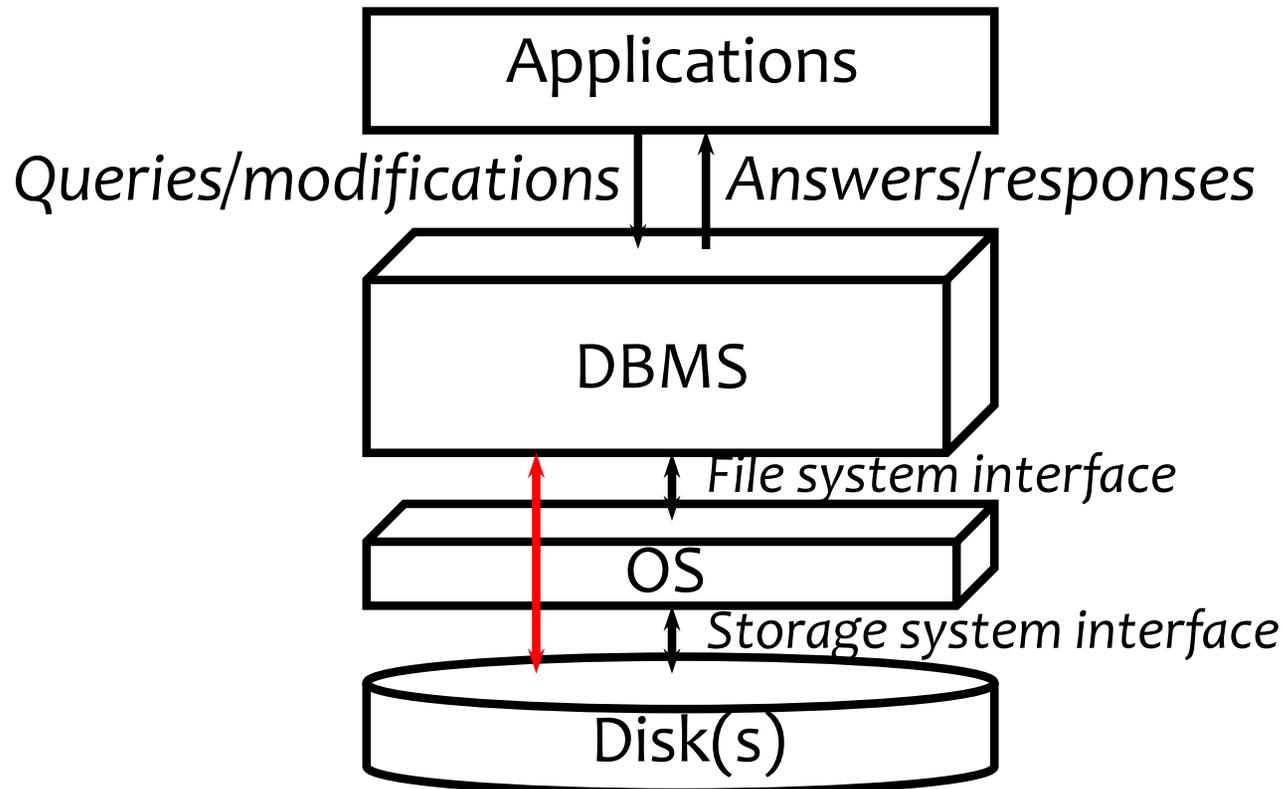
Recovery in DBMS

- 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 instruction
- 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 ($\geq 99.999\%$ 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...
 - This course will look beyond relational databases



Use of AYBABTU inspired by Garcia-Molina

Image: <http://upload.wikimedia.org/wikipedia/en/0/03/Aybabtu.png>

Course components

- Relational databases
 - Relational algebra, database design, SQL, app programming
- XML
 - Data model and query languages, app programming, interplay between XML and relational databases
- Database internals
 - Storage, indexing, query processing and optimization, concurrency control and recovery
- Advanced topics (TBD)
 - Data warehousing and data mining, Web search and indexing, parallel data processing/MapReduce, etc.

Announcements (Tue. Aug. 29)

- Email me for registration questions
- Thursday
 - Andrew + others will lead the class through the steps of setting up VM for the class on your laptop
 - Bring your laptop, fully charged!
- I will be back to talk about relational algebra next Tuesday!