Introduction

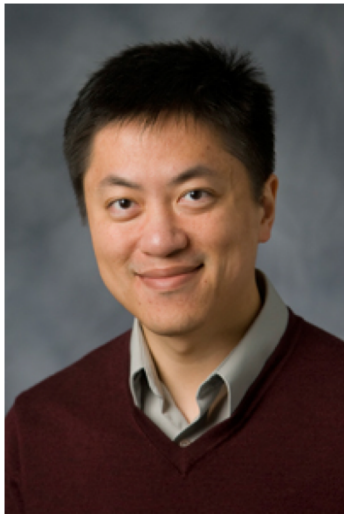
Introduction to Databases

CompSci 316 Fall 2019

DUKE
COMPUTER SCIENCE

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



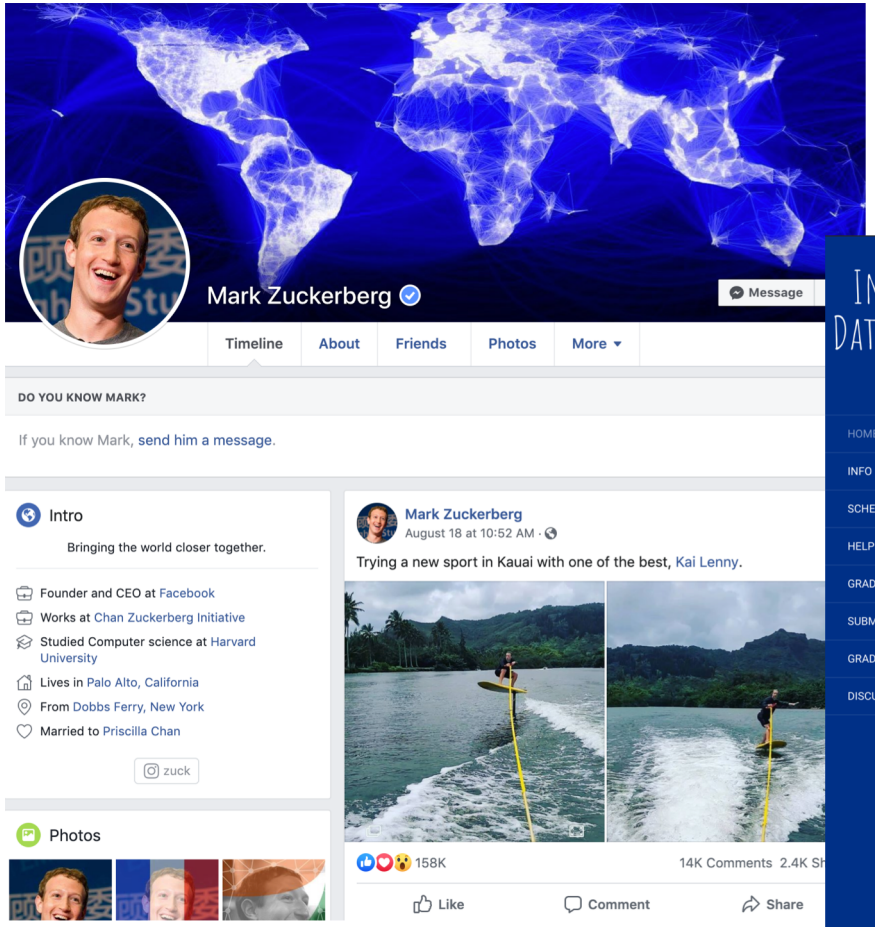
- Graduate TAs:
  - Tiangang Chen
  - Yanlin Yu
  - MS students in Computer Science
- UTAs: *still forming the team*

What comes to your mind…

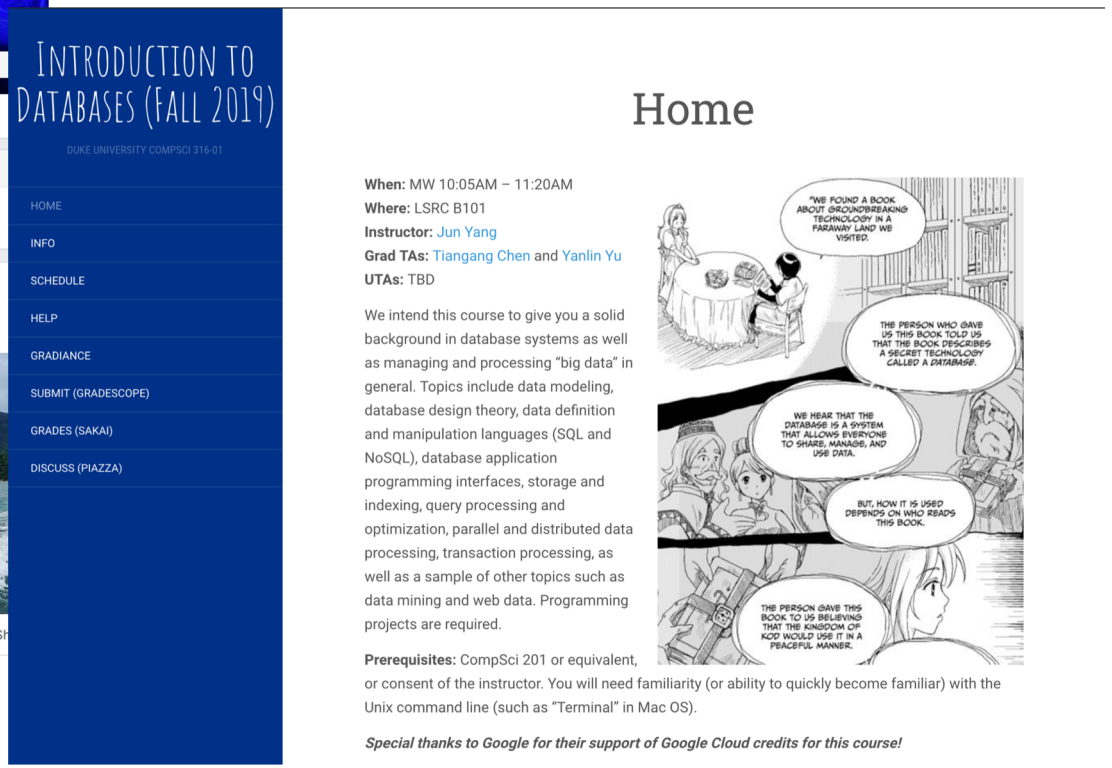
… when you think about “databases”?

https://www.bing.com/th?id=OIP.EweafL8YeA09QWB4V53HuAHaFj&pid=Api&rs=1
https://www.tibco.com/blog/wp-content/uploads/2017/11/graph-database.png

But these use databases too…



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



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

Data → fun and profit

FiveThirtyEight

Politics Sports Science & Health Economics **Culture**

APR. 24, 2019, AT 10:38 AM

The Man Who Solved 'Jeopardy!'

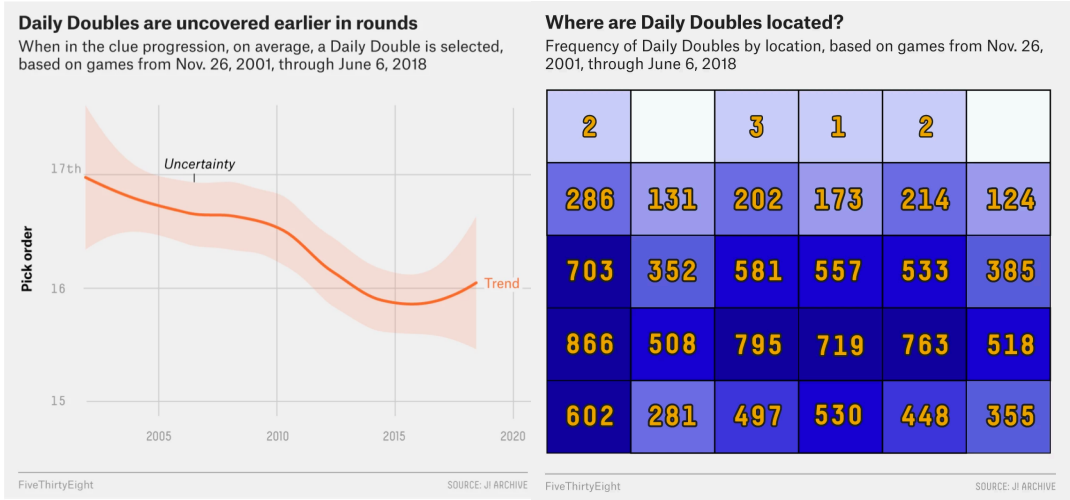
James Holzhauer has taken the game to its logical conclusion.

By Oliver Roeder

Filed under Jeopardy!

JEOPARDY PRODUCTIONS, INC.

UPDATE (April 26, 2019, 2:45 p.m.): James Holzhauer has won 16 consecutive "Jeopardy!" games, including eight of the 10 richest games in the show's history.



And what did it take to unseat this champion?

We Found "Jeopardy!" Kingslayer Emma Boettcher's Thesis on Data Mining

Did Boettcher's data work help her dethrone reigning champ James Holzhauer?

FOSTER KAMER | JUNE 4TH 2019

"… Boettcher's 2016 paper, titled "Predicting the Difficulty of Trivia Questions Using Text Features," was her master's thesis for the School of Information and Library Science at UNC Chapel Hill …"

https://fivethirtyeight.com/features/the-man-who-solved-jeopardy/
https://futurism.com/jeopardy-emma-boettcher-ai-james-holzhauer

Data → science

FiveThirtyEight

Politics Sports **Science & Health** Economics Culture

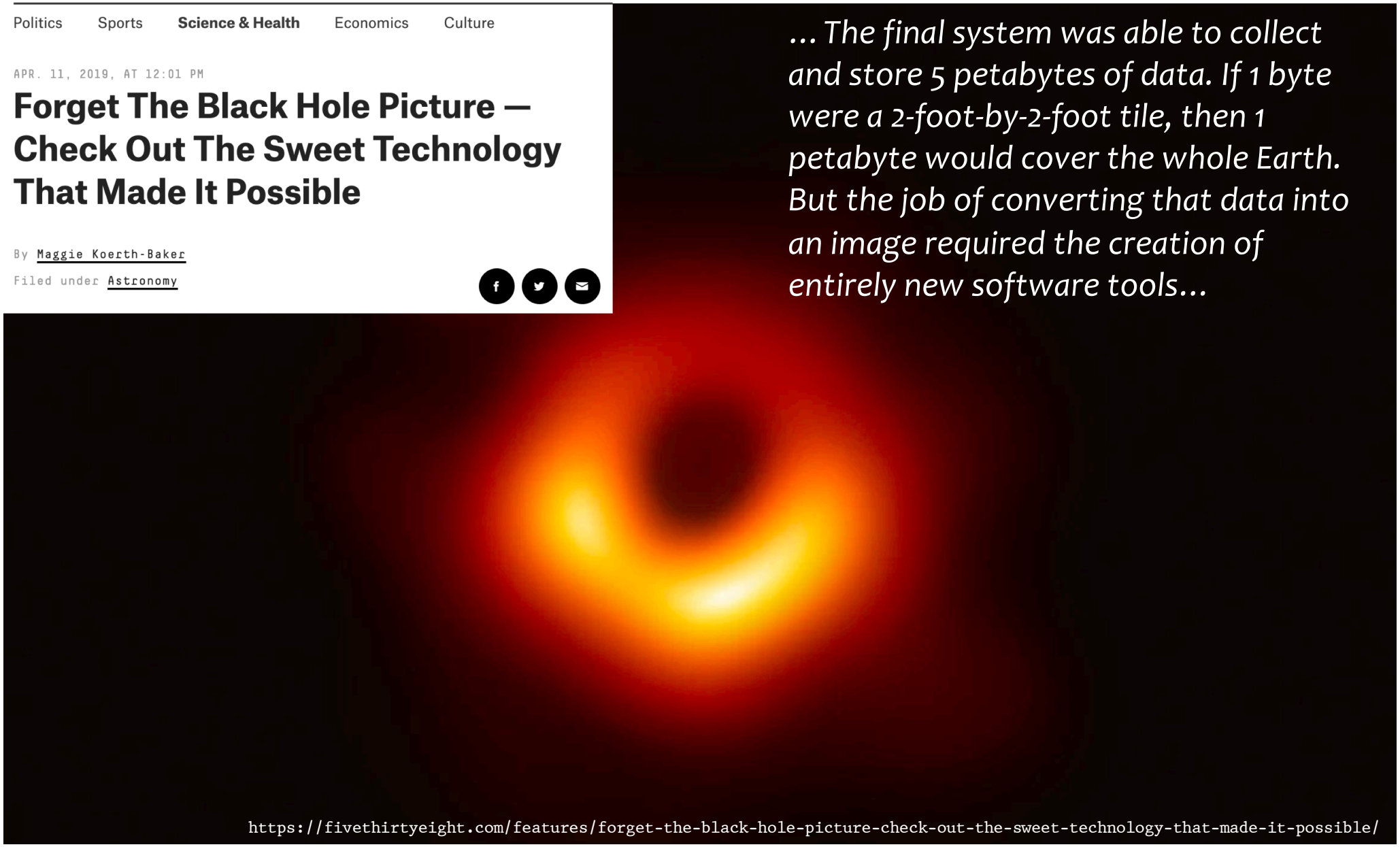
APR. 11, 2019, AT 12:01 PM

Forget The Black Hole Picture — Check Out The Sweet Technology That Made It Possible

By Maggie Koerth-Baker

Filed under Astronomy

… 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://fivethirtyeight.com/features/forget-the-black-hole-picture-check-out-the-sweet-technology-that-made-it-possible/

Data → power

… In total, the hacked data includes records from 19 states… Researchers at Anomali Labs believe that the data includes records for more than 35 million U.S. voters… Voter Data for each state appears to be priced according to the hackers' anticipated demand. The Texas file appears to be the largest with information on 14 million voters, but it's nowhere near the most expensive. Wisconsin -- with fewer than half as many records -- was priced at $12,500. The seller will bundle all 19 for $42,200…

1,093 views | Oct 16, 2018, 11:30am

Millions of Voter Records Are For Sale On Hacker Forums

Lee Mathews Senior Contributor
Cybersecurity
Observing, pondering, and writing about tech. Generally in that order.

There are just under three weeks until the 2018 midterm elections. It's not great timing, then, to learn that a user on a hacking forum is selling voter data on millions of Americans.

A hand placing a voting slip into a ballot box

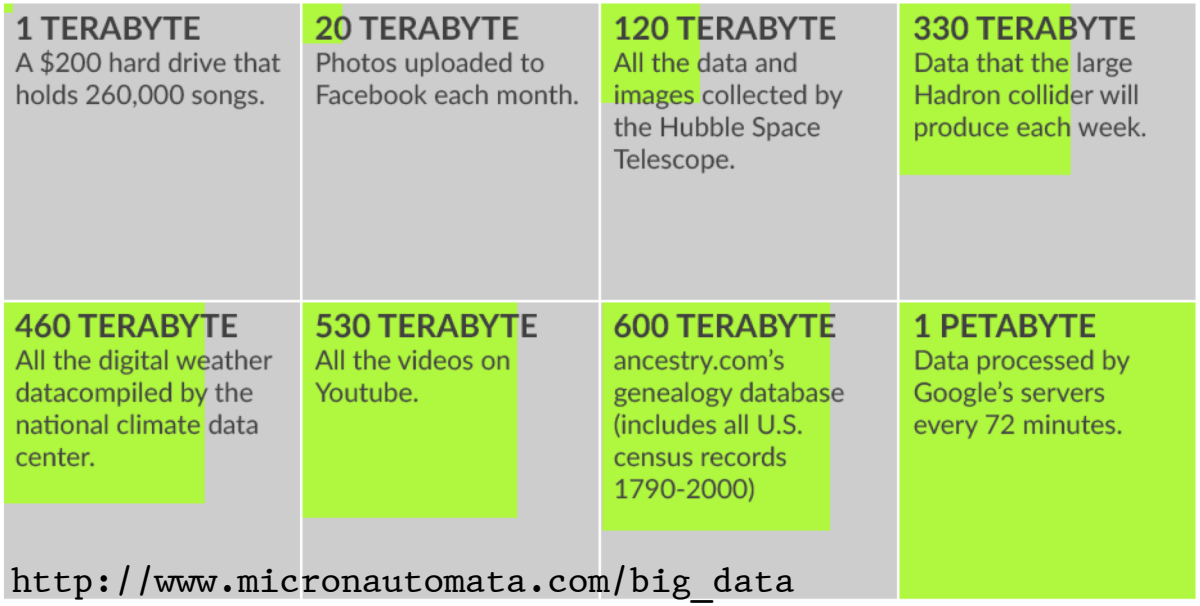
https://www.forbes.com/sites/leemathews/2018/10/16/millions-of-voter-records-are-for-sale-on-hacker-forums/

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



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: `http://sites.duke.edu/compsci316_01_f2019/`
  - 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
- 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

- ~A dozen Gradiance exercises (10%)
  - Immediately and automatically graded
- Four homework assignments (25%)
  - Written and programming problems; submit through Gradescope
- 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

- 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
- 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

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
  - Gautam Hathi, Joy Patel, Seon Kang, Zoey Zou, 2016
- 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
  - 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
  - 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

http://www.yummymummyclub.ca/sites/default/files/styles/large/public/field/image/teaching_kids_creative_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!

```
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

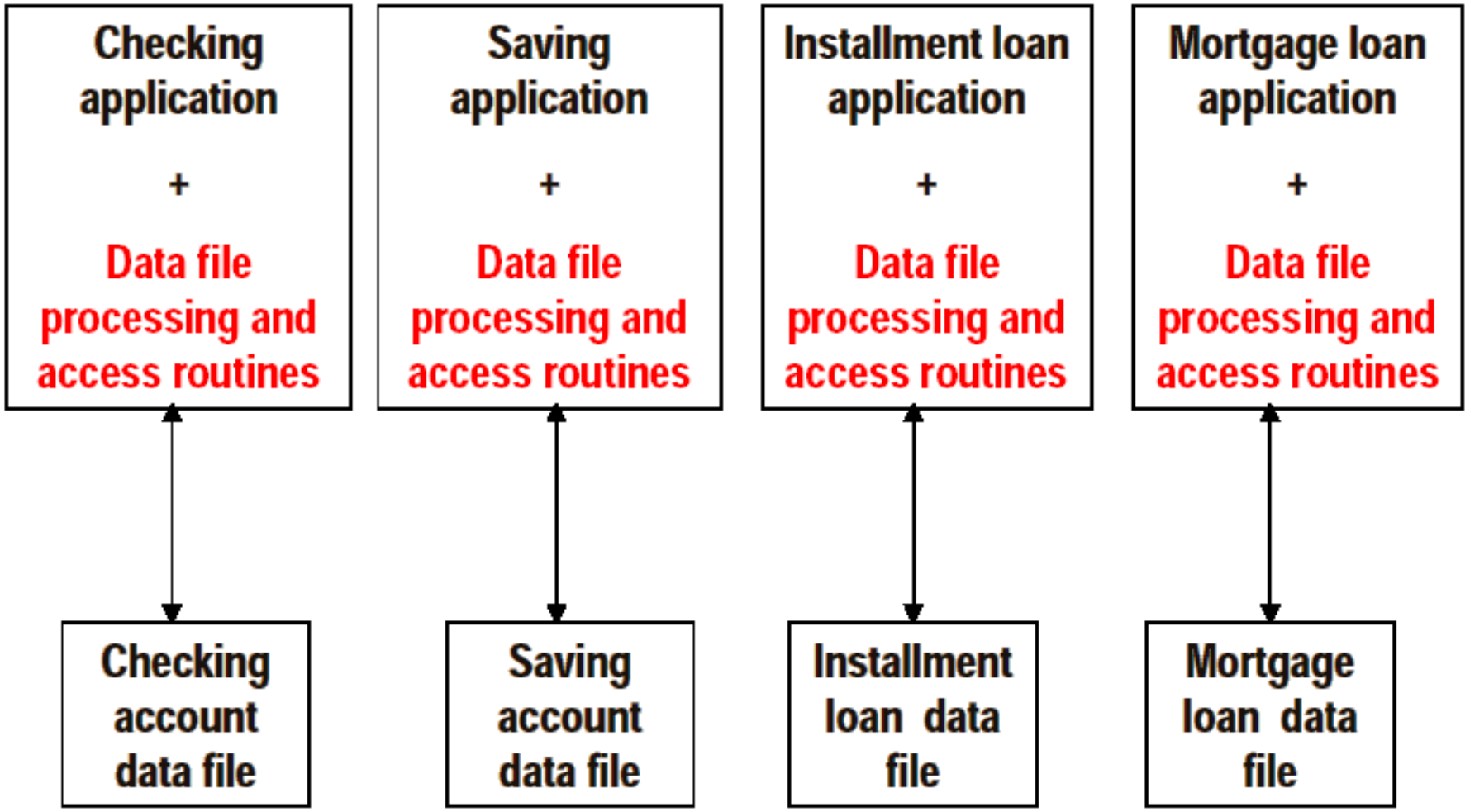
- Tens of thousands of accounts are not Homer's
  - ☞Cluster accounts by owner's initial: those owned by "A..." go into file A; those owned by "B..." go into file B; etc. → decide which file to search using the initial
  - ☞Keep accounts sorted by owner name → binary search?
  - ☞Hash accounts using owner name → compute file offset directly
  - ☞Index accounts by owner name: index entries have the form ⟨*owner_name*, *file_offset*⟩ → 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

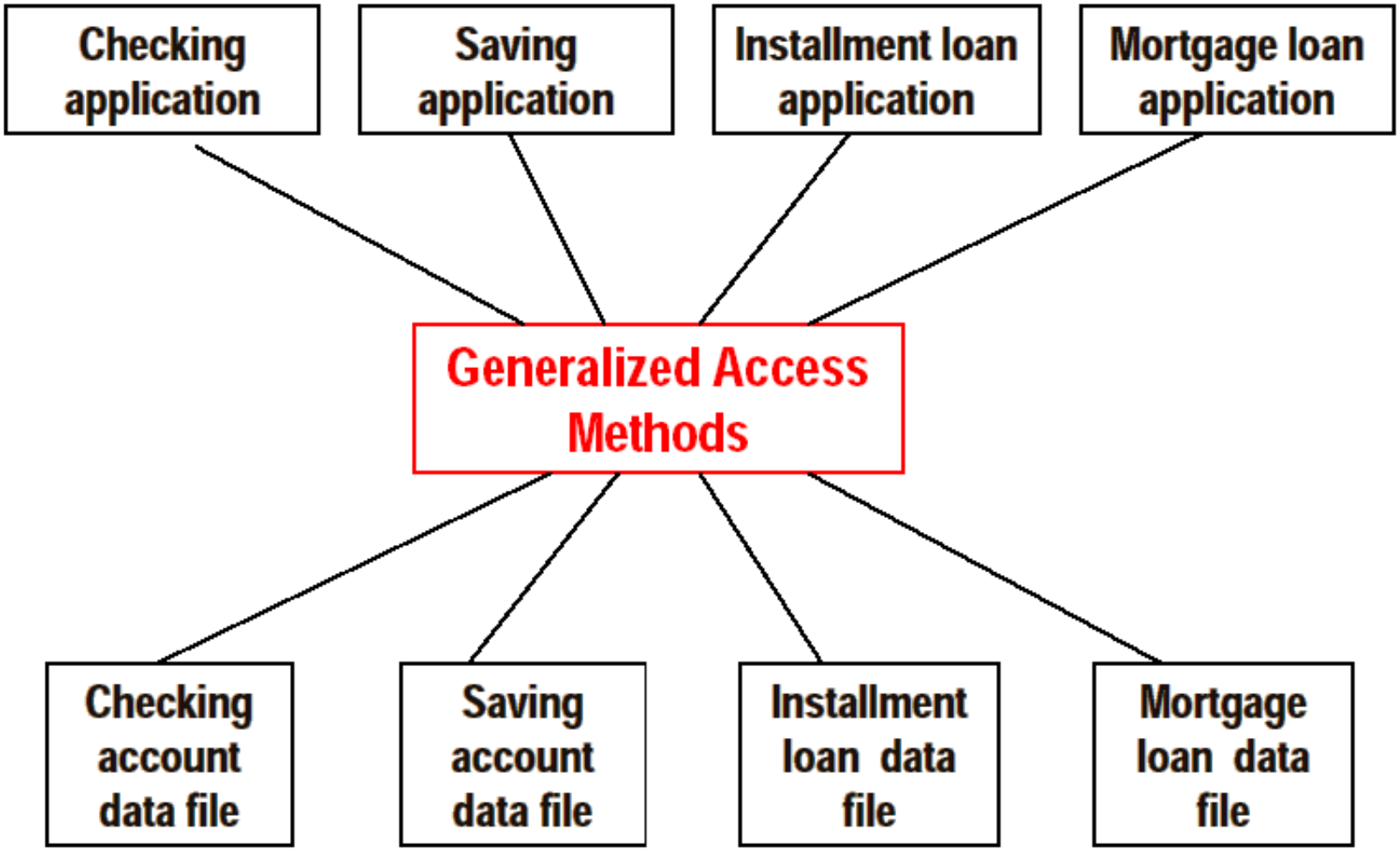
- There are many techniques—not only in storage and query processing, but also in concurrency control, recovery, etc.
- These techniques get used over and over again in different applications
- Different techniques may work better in different usage scenarios

The birth of DBMS – 1



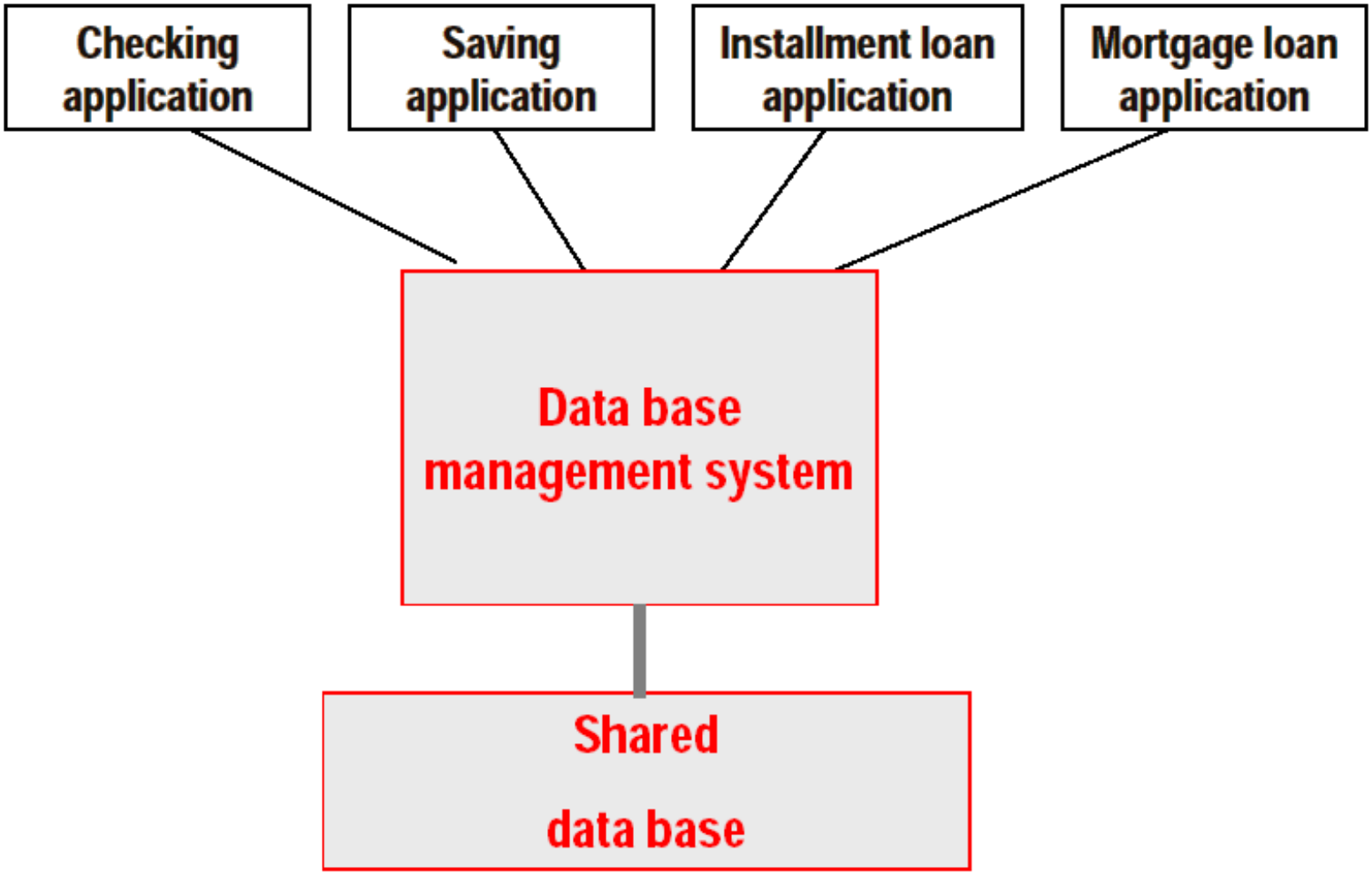
From Hans-J. Schek's *VLDB* 2000 slides

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 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

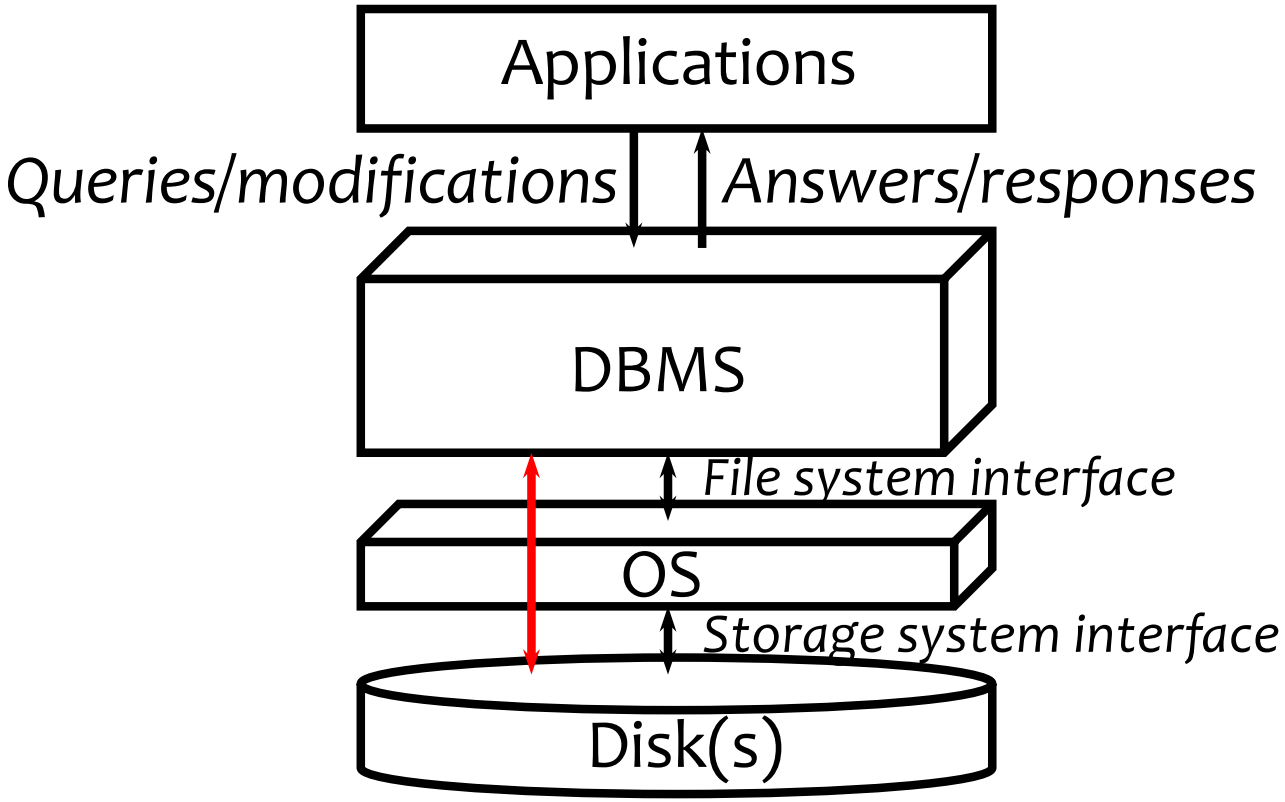
```
decrement the balance of account X by $100;
increment the balance of account Y by $100;
```

- 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 (≥ 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
- Semi-structured data
  - Data model and 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, data warehousing and data mining, Web search and indexing, etc.

Announcements (Mon. Aug. 26)

- No office hours today; please talk to me after lecture
- Need a spot in the class?
  - Keep in mind that 316 will also be offered in Spring 2020!
  - We will go down the wait list in order (on Friday noon) to give permission numbers
- Thursday: we will do relational algebra—the first of many query languages we shall learn
  - I will be out of town, but TA will conduct the class with my pre-recorded lecture
  - More info on course VM setup and Google credits via email
  - Homework 1 and the first Gradiance exercise will be assigned