Introduction

Introduction to Databases
CompSci 316 Spring 2017
Welcome to

CompSci 316: Introduction to Database Systems!!
Spring 2017

Acknowledgement: Thanks to Prof. Jun Yang for the course material of CompSci 316!
First things first!

• Please occupy the first few rows, i.e. pack from the front

• Please raise your hand and let me know whenever you cannot hear me clearly

• Ask me to slow down or repeat any time

• Interruptions and questions are always welcome!
About us: instructor

• Instructor: Sudeepa Roy
  • Assistant Professor in Duke Computer Science since Fall 2015
  • A proud member of “Duke Database Devils” group 😊
    https://sites.duke.edu/duke_dbservice/
  • PhD. UPenn, Postdoc: U. of Washington
  • Research interests: data management, database theory, data analysis, causality and explanations, uncertain data, data provenance, crowdsourcing, ....
About us: Graduate TAs

• Graduate TA: Junyang Gao
  • PhD student in Computer Science
  • Working on fact checking, time series data, and query optimization

• Graduate TA: Yuhao Wen
  • PhD student in Computer Science
  • Working on data-intensive interactive systems
About us: UTAs

Anh  CompSci 316 veteran!

Bill  Both CompSci 516 veterans!

Wilson
What comes to your mind...

...when you think about “databases”?

http://www.quackit.com/pix/database/tutorial/dbms_sql_server.gif
But these use databases too...

Facebook uses MySQL to store posts, for example

WordPress uses MySQL to manage components of a website (pages, links, menus, etc.)
Data → Gold (ok, Bronze)

... 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 → Profit and Fun

‘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

... 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

"What is the EU?" is the second top UK question on the EU since the #EURefResults were officially announced.

A database with information on all American voters... might go for about $270,000, according to one marketing firm consulted by researcher Chris Vickery...
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*

<table>
<thead>
<tr>
<th>1 TERABYTE</th>
<th>20 TERABYTE</th>
<th>120 TERABYTE</th>
<th>330 TERABYTE</th>
</tr>
</thead>
<tbody>
<tr>
<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>530 TERABYTE</td>
<td>600 TERABYTE</td>
<td>1 PETABYTE</td>
</tr>
<tr>
<td>All the digital weather data compiled by the national climate data center.</td>
<td>All the videos on Youtube.</td>
<td>ancestry.com’s genealogy database (includes all U.S. census records 1790-2000)</td>
<td>Data processed by Google’s servers every 72 minutes.</td>
</tr>
</tbody>
</table>

“An estimate for 2020 is approximately 40 zettabytes of data on the web, which equals a whopping 43 trillion gigabytes.”

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 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_s2017/](http://sites.duke.edu/compsci316_01_s2017/)
  • Course info; tentative schedule and reference sections in the book; lecture slides, assignments, help docs, ...


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

• Q&A on Piazza – use and check frequently – feel free to post anonymously

• Grades, sample solutions on Sakai

• Watch your email for announcements

• Office hours posted on the webpage
Grading

>= 30% of the class  A- / A / A+
the next >= 40% of the class  B- / B / B+
the next <= 30% of the class  Cs, Ds, Fs

• Adjustable “curves”
• Note >=, <=!
• i.e. Scale may be adjusted downwards (i.e., grades upwards) if, for example, the class performance is relatively uniform for a group
• Scale will not go upwards
• Potentially the average of the class can be higher!
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 by the third week of class
  • Up to 4 students in each group

• 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 earlier years

• **RA**: next-generation relational algebra interpreter, 2015
  • Kevin Do, Michael Han, Jennie Ju, Jordan Ly
  • You may get to try it out for Homework #1!

• **wikiblocks** ([https://vimeo.com/147680387](https://vimeo.com/147680387)), 2015
  • Brooks Mershon, Mark Botros, Manoj Kanagaraj, Davis Treybig
  • Automatically finds relevant interactive visualizations when you are browsing a Wikipedia page
Projects from earlier years

• **SMSmart** (4.1 stars on Google Play): search/tweet/Yelp without data by sms offline
  • 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 Schedulator: ditch ACES—plan visually!
  • Alex Beutel, 2008

• SensorDB: manage/analyze sensor data from forest
  • Ashley DeMass, Jonathan Jou, Jonathan Odom, 2007
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!

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

Generalized Access Methods

Checking account data file
Saving account data file
Installment loan data file
Mortgage loan data file

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

• Aside: *Four Turing Awards in databases so far!*

• 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;
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
- In reality, most big databases today are distributed
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

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

• XML
  • Data model and query languages, app programming, interplay between XML and relational databases

• Advanced topics (TBD)
  • Data warehousing and data mining, parallel data processing/MapReduce, NOSQL etc.
Announcements (Wed. Jan 11)

• Check your account on sakai
• Enroll to piazza (follow the “Discussion” link from course webpage)
• Enroll to gradiance with class token 7E9C31D0 (follow the “Gradiance” link from course webpage)
• No class on Monday 01/16 – MLK Day holiday!
• Next Wednesday 01/18
  • Our first language of the semester—relational algebra!
  • Homework #1 to be assigned