Your Instructor: Luis Gravano

- Ph.D. in Computer Science, Stanford U.
- Professor, Computer Science Department (at Columbia U. since Fall 1997)
- At Google: Senior Research Scientist (2001), Visiting Faculty Researcher (2018-19)

- Research interests: Databases, Web Search, Information Extraction, Social Media
Class Resources

- Class website: http://www.cs.columbia.edu/~gravano/cs4111
- Discussion board: Piazza, which you can access from CourseWorks, at https://courseworks.columbia.edu/
- Announcements from class staff: on CourseWorks
Your Instructor: Luis Gravano

- [gravano@cs.columbia.edu](mailto:gravano@cs.columbia.edu)

- Office hours (online):
  - Mondays and Fridays, 10:30-11:30 a.m. ET
  - Tuesdays and Thursdays, 2:30-2:45 p.m. ET, for post-lecture questions
  - By appointment by email
Your TAs

- Sara Bernstein
- Ivy Chen
- Noah Huber-Feely
- Ruoyi Nie
- Jonathan Rich
- ...

- All office hours are held online
- Office hours and their links are on a Google Calendar in CourseWorks “Announcement”
Class Information: Prerequisites

COMS W3134 – Data Structures in Java, COMS W3136 – Essential Data Structures in C/C++, or COMS W3137 – Data Structures and Algorithms (equivalent courses taken elsewhere are acceptable as well)

You need permission from the instructor if you don’t have the prerequisites
Class Information: Lectures

- Tuesdays and Thursdays, 1:10-2:25 p.m. ET
- Fully online: Zoom link for lectures available on CourseWorks, in “Zoom Class Sessions” section
- Lecture recordings available within 24 hours on CourseWorks, in “Video Library” section
Lectures on Zoom

- Mute yourself but keep your video on during lectures as much as possible
- Ask questions in the Zoom chat or by using Zoom’s “raise hand” function
  - Questions are welcome and encouraged!
- Technical difficulties: if instructor gets disconnected during lecture, or Zoom is down, try to reconnect for 5 minutes; if problem persists after 5 minutes, check CourseWorks “Announcements” section for directions
Grading Information

- **Midterm** (Thu Oct 29, during lecture time): 15%
- **Final** (Tue Dec 22, 1:10-3:10 p.m. ET, cumulative): 30%
- **Homework assignments** (4, all equally weighted): 15%
- **Projects** (2): 40%
  - Project 1: 30%, Project 2: 10%
- Midterm and final are online, closed book, closed notes; alternative accommodations available for students in time zones far from New York’s
- Median grade will be a B+ or slightly higher
- Alternative or make-up exams will **not** be given
- All homework assignments are equally weighted
- Project 1 has higher weight than Project 2
Homework

• To be fair to all students in the class, I will grant no extensions or exceptions.
• Instead, you have three grace late days for homework that you can use as you wish throughout the semester; weekends and university holidays are not counted.
• After using all grace days, you will get a 25% grade deduction for each additional late day.

Check full details on website.
2 Projects, in Teams of 2 Students

- You will do the projects on the Google Cloud platform
- You will have more-than-enough free credit through individual codes that I will distribute once enrollments stabilize
- Projects will have a non-programming option
- If you follow the programming option, which I strongly recommend, you will program in Python (only language option):
  - Python is much easier to work with than Java for our database projects
  - Python is a great, easy-to-learn, widely used language
  - If you are fluent in Java, you will be able to easily learn the (not-so-deep) level of Python needed for our projects

- Project 1: model and build an application of your choice on top of a database system, using “traditional” relational database features
- Project 2: expand Project 1 to use substantial, advanced database system features

More details announced soon; please be patient and wait until projects announced
Projects (cont.)

• To be fair to all students in the class, I will grant no extensions or exceptions for project submission.

• Instead, you have three grace late days total for projects that you can use as you wish throughout the semester; weekends and university holidays are not counted.

• After using all grace days, you will get a 25% grade deduction for each additional late day.

Check full details on website.
Collaboration Policy

- Please check "Collaboration and Academic Honesty" page from the main webpage for the class
- Homework and exams are to be done individually
- Projects are done in teams; no collaboration between different teams
- We will not tolerate cheating, which would be wrong and unfair to the rest of the class. Check the CS Department policies and procedures regarding academic honesty at http://www.cs.columbia.edu/education/honesty; they fully apply to this course.
- Contact the instructor right away if you have any questions
Optional Textbook


Textbook is optional: lectures will cover all material needed for homework assignments and exams

Textbook homepage has useful resources: https://www.db-book.com/db7/
Contests and Rewards

• “A+ letter-and-lunch” reward:
  Students who earn an A+ in the course will get:
  • A personal letter of congratulations from the instructor
  • An invitation to join the instructor and the other A+ students
    for lunch at the Columbia Faculty House (pandemic-permitting)
  These students will also be top candidates for cs4111 TA
    positions in the future

• Project 1 contest:
  • Four best projects chosen as contest winners
  • If you win:
    • You will have the option to discuss and demonstrate your
      project in class
    • You will get a 10% boost in your Project 1 grade
Ongoing Feedback

• Don’t wait until the end-of-semester course evaluations to complain or give feedback on how to improve course (it’s too late then!)

• Talk to me early on during my office hours or send me email with your concerns and suggestions, or ask a TA to forward them to me
Thanks to Raghu Ramakrishnan, Johannes Gehrke, and our own Eugene Wu for some of the slides!
What Is a Database and a Database Management System (DBMS)?

- A database is a generally large, integrated collection of data that models a real-world enterprise
  - Entities (e.g., students, courses)
  - Relationships (e.g., Jane Smith is taking cs4111)

- A database management system (DBMS) is a software system designed to store, manage, and interact with databases
Why Use a DBMS?

- Data independence and efficient access
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes
Why Study Databases?

• Shift from computation to information
• Data sets increasing in diversity and volume
  • The Web, online activity and commerce, social media, …
  • ... need for DBMS exploding
• DBMS encompasses most of CS
  • OS, languages, theory, AI, machine learning, natural language processing, multimedia, …
Why Study Databases?

• Most structured information on the web lives in databases
  • Projects 1 and 2 will give you the opportunity to understand their potential for data and a domain of your interest

• Databases are critical to organize, query, and perform data analysis (e.g., data mining) of scientific, business and financial, environmental, health data, and much more

• Also extremely helpful to organize large-scale experimental results and data

• Other types of data repositories, such as text content, covered in COMS E6111
Why are DBMSs Necessary?

Consider a bank with a simple “relation” with account information:
Why are DBMSs Necessary?

Consider a bank with a simple “relation” with account information:

<table>
<thead>
<tr>
<th>accountNo</th>
<th>balance</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>$1000</td>
<td>savings</td>
</tr>
<tr>
<td>54321</td>
<td>$250</td>
<td>checking</td>
</tr>
<tr>
<td>12345</td>
<td>$150</td>
<td>checking</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Users can withdraw money from an account
- Users can transfer money between accounts
- Bank can ask queries such as **Q1**: “list all accounts with a balance over $10K”
Why not implement all this with a file system with some ad-hoc software?
How About Efficiency?
Providing “Transactional Guarantees”?
Expected Behaviors if You and Your Spouse Withdraw Money or Pay a Bill on Same Account Simultaneously?
Yes, We Could Write All Code for Our Applications Ourselves ...
Yes, We Could Write All Code for Our Applications Ourselves ... 

... in a few decades!

Luckily, this functionality (queries, query processing, transaction processing) is common across many applications, so we can factor it out.
Database Courses at Columbia
(This course) COMS W4111-Introduction to Databases
Prerequisites: CS3134, CS3136, or CS3137

- The Entity-Relationship Model
- The Relational Model
- The Relational Algebra
- SQL: Queries, Constraints, Triggers
- Embedded SQL, Cursors, SQL APIs
- Schema Refinement and Normal Forms
- Object-Relational DBMS: Database Design
- Introduction to Query Processing and Optimization
- Introduction to Transaction Processing
COMS W4112-Database System Implementation

Prerequisites: CS4111; fluency in Java or C++; recommended: CS3827

• Storage Methods and Indexing
• Query Processing and Optimization for 1NF Relations, including External Sorting
• Materialized View Maintenance, Selection, and Use in Query Optimization
• Query Processing and Optimization for ORDBMSs
• Transaction Processing and Recovery
• Parallel and Distributed Databases: Query Processing and Optimization
• Parallel and Distributed Databases: Transaction Processing
• Performance Considerations Beyond I/Os
COMS E6111-Advanced Database Systems
Prerequisites: CS4111; fluency in Java or Python

- Information Retrieval
- Information Extraction
- Web Search
- Data Mining
- Data Warehousing, OLAP, Decision Support
- Time Series Analysis and Mining
- Spatial Data Management
- ...
Overview of Database-Related Issues
Data Models

• A data model is a collection of concepts for describing data

• The relational model of data is the most widely used model today
  • Main concept: relation, basically a table with rows and columns
  • Every relation has a schema, which describes the columns, or fields

<table>
<thead>
<tr>
<th>accountNo</th>
<th>balance</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>$1000</td>
<td>savings</td>
</tr>
<tr>
<td>54321</td>
<td>$250</td>
<td>checking</td>
</tr>
<tr>
<td>12345</td>
<td>$150</td>
<td>checking</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Levels of Abstraction

- Many views, single conceptual (logical) schema, and physical schema
  - Views describe how users see the data
  - Conceptual schema defines logical structure
  - Physical schema describes files and indexes used

- Schemas are defined using a data definition language (DDL)
- Data is modified and queried using a data manipulation language (DML)
Example: University Database

- **Conceptual schema:**
  - Students(sid:string, name:string, login:string, age:integer, gpa:real)
  - Courses(cid:string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)
Example: University Database

• Conceptual schema:
  • Students(sid:string, name:string, login:string, age:integer, gpa:real)
  • Courses(cid:string, cname:string, credits:integer)
  • Enrolled(sid:string, cid:string, grade:string)

• Physical schema:
  • Relations stored as unordered files
  • Index on first column of Students
Example: University Database

- **Conceptual schema:**
  - Students(sid:string, name:string, login:string, age:integer, gpa:real)
  - Courses(cid:string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)

- **Physical schema:**
  - Relations stored as unordered files
  - Index on first column of Students

- **External schema (view):**
  - Course_info(cid:string, cname:string, enrollment:integer)
Data Independence

- Applications insulated from how data is structured and stored
- **Logical data independence**: Protection from changes in logical structure of data
- **Physical data independence**: Protection from changes in physical structure of data

- One important benefit of using a DBMS
Concurrent execution of user programs is essential for good DBMS performance.

Interleaving actions of different user programs can lead to inconsistency: what if check is cleared while account balance is being computed?

DBMS ensures such problems don’t arise: users can pretend they are using a single-user system.
Transaction: An Execution of a Database “Program”

- Key concept: transaction, an atomic sequence of database actions
  - Example: Transferring funds from one account to another, which requires multiple operations over bank database

- DBMS guarantees important properties of transactions automatically, without user involvement:
  - Preserving consistency of database contents
  - Handling crashes gracefully, without ever losing any data
  - Allowing concurrent execution in a controlled manner that doesn’t create any problems
Summary

• DBMS used to maintain, query large databases
• Benefits include recovery from system crashes, concurrent access, quick and flexible application development, data integrity and security
• Levels of abstraction give data independence

• Databases is one of the **broadest, most useful** research and development areas in Computer Science