COMS W4111.001
Introduction to Databases
Spring 2017

Computer Science Department
Columbia University

Your Instructor: Luis Gravano

• Ph.D. in Computer Science, Stanford U.
• Professor, Computer Science Department
  (at Columbia U. since Fall 1997)
• Senior Research Scientist at Google in 2001
  (on leave from Columbia U.)
• Research interests: Databases, Web Search,
  Information Extraction, Social Media
Class Resources

• Class web page:
  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

Various addresses and numbers:
  http://www.cs.columbia.edu/~gravano
  gravano@cs.columbia.edu
  706 Schapiro CEPSR
  1-212-939-7064

Office hours:
  Wednesdays, 9:30-11:30 a.m.
  By appointment by email
Your IAs

- Atif Ahmed
- Terra Blevins
- Min Fan
- Jibben Hillen
- Aayush Mudgal
- ...

- All IA office hours are held in the CS IA Room (directions from class web page)
- IA office hours already posted on class web page

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)

- Fluency in Java

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

- Tuesdays and Thursdays
- 2:40-3:55 p.m.
- 501 Northwest Corner Building

Grading Information

- **Midterm** (Thu Mar 9, in class, closed book): 25%
- **Final** (Thu May 11, 1:10-4:00 p.m., closed book): 40%
- **Homework assignments** (4, all equally weighted): 15%
- **Projects** (2): 20%
  - Project 1: 15%, Project 2: 5%

- 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 web site.

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 (or "minimal 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
  - You are fluent in Java, so 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 web site.

Collaboration Policy

- Please check “Collaboration and Academic Honesty” page from the main web page 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.
Textbook

Raghu Ramakrishnan, Johannes Gehrke:
• On reserve in Science and Engineering Library

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.
  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!)
• Come see me early on during my office hours or send me email with your concerns and suggestions, or use an IA to forward them to me.

Thanks to Raghu Ramakrishnan Johannes Gehrke, and our own Eugene Wu for slides and additional supporting material!
What Is a DBMS?

- A very large, integrated collection of data.
- Models real-world enterprise.
  - Entities (e.g., students, courses)
  - Relationships (e.g., Jane Smith is taking cs4111)
- A Database Management System (DBMS) is a software package designed to store and manage 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
  - Examples?
  - Project 1 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
Database Courses at Columbia

(This course) COMS W4111-Introduction to Databases
Prerequisites: CS3134, CS3136, CS3137; fluency in Java

- Course Overview, Introduction to Database Management Systems (DBMSs)
- The Entity-Relationship Model
- The Relational Model
- The Relational Algebra
- SQL: Queries, Constraints, Triggers
- Embedded SQL, Cursors, SQL APIs
- Object-Relational DBMS: Database Design
- Schema Refinement and Normal Forms
- Introduction to Query Processing and Optimization
- Introduction to Physical Database Tuning
- 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
• Web Search
• Data Mining
• Data Warehousing, OLAP, Decision Support
• Spatial Data Management
• Information Extraction
• ...
Overview of Database-Related Issues

Data Models

- A **data model** is a collection of concepts for describing data
- A **schema** is a description of a particular collection of data, using a given data model
- 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
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 the files and indexes used.
- Schemas are defined using DDL; data is modified and queried using 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)
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External schema (view):
  - Course_info(cid: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 of the most important benefits of using a DBMS!

Concurrency Control

• Concurrent execution of user programs is essential for good DBMS performance. Because disk and memory accesses are frequent, and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently.
• Interleaving actions of different user programs can lead to inconsistency: e.g., 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 DB Program

- Key concept is transaction, which is an atomic sequence of database actions (reads and writes).
- Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
  - Users can specify some simple integrity constraints on the data, and the DBMS will enforce these constraints.
  - Beyond this, the DBMS does not really understand the semantics of the data (e.g., it does not understand how the interest on a bank account is computed).
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

Scheduling Concurrent Transactions

DBMS ensures that execution of \{T_1, \ldots, T_n\} is equivalent to some serial execution \(T_1' \ldots T_n'\).

- Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (Strict 2PL locking protocol.)
- Idea: If an action of \(T_i\) (say, writing X) affects \(T_j\) (which perhaps reads X), one of them, say \(T_{i_i}\), will obtain the lock on X first and \(T_j\) is forced to wait until \(T_i\) completes; this effectively orders the transactions.
- What if \(T_i\) already has a lock on Y and \(T_j\) later requests a lock on Y? (Deadlock!) \(T_i\) or \(T_j\) is aborted and restarted!
Ensuring Atomicity

- **DBMS ensures atomicity** (all-or-nothing property) even if system crashes in the middle of a transaction.
- **Idea:** Keep a log (history) of all actions carried out by the DBMS while executing a set of transactions:
  - **Before** a change is made to the database, the corresponding log entry is forced to a safe location. *(WAL protocol; OS support for this is often inadequate.)*
  - **After** a crash, the effects of partially executed transactions are undone using the log. (Thanks to WAL, if log entry wasn’t saved before the crash, corresponding change was not applied to database!)

---

The Log

- The following actions are recorded in the log:
  - Ti writes an object: old value and new value.
    - Log record must go to disk before the changed page.
  - Ti commits/aborts: a log record indicating this action.
  - Log records chained together by transaction id, so it’s easy to undo a specific transaction.
  - Log is often duplexed and archived on “stable” storage.
  - All log related activities are handled transparently by the DBMS.
Databases Make these Folks Happy...

- End users and DBMS vendors
- Database application programmers
- Database administrator (DBA)
  - Designs logical and physical schemas
  - Handles security and authorization
  - Guarantees data availability, crash recovery
  - Tunes database as needs evolve
- Must understand how a DBMS works!

Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.

These layers must consider concurrency control and recovery
Summary

• DBMS used to maintain, query large data sets.
• Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
• Levels of abstraction give data independence.
• A DBMS typically has a layered architecture.
• DBAs hold responsible, critical jobs.
• Databases is one of the broadest, most exciting research and development areas in CS.