CS411: Database Systems

01: Introduction

Abdu Alawini
Teaching Staff

• Abdu Alawini
  – Teaching Assistant Professor
  twitter: @aalawini
  web: http://www.alawini.com

• Research interests:
  – Database Systems
  – Scientific Data Management
  – Computer Science Education
## Teaching Staff: TAs

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Photo" /></td>
<td>Zhiran Sun</td>
<td>Himel Dev</td>
</tr>
<tr>
<td><strong>Zhiran Sun</strong></td>
<td><strong><a href="mailto:zhirans2@illinois.edu">zhirans2@illinois.edu</a></strong></td>
<td><strong><a href="mailto:hdev3@illinois.edu">hdev3@illinois.edu</a></strong></td>
</tr>
<tr>
<td><img src="image2" alt="Photo" /></td>
<td>Aravind Sankar</td>
<td></td>
</tr>
<tr>
<td><strong>Aravind Sankar</strong></td>
<td><strong><a href="mailto:asankar3@illinois.edu">asankar3@illinois.edu</a></strong></td>
<td></td>
</tr>
<tr>
<td><img src="image3" alt="Photo" /></td>
<td>Tianyu Ao</td>
<td></td>
</tr>
<tr>
<td><strong>Tianyu Ao</strong></td>
<td><strong><a href="mailto:tianyua2@illinois.edu">tianyua2@illinois.edu</a></strong></td>
<td></td>
</tr>
<tr>
<td><img src="image4" alt="Photo" /></td>
<td>Stuart Zhu</td>
<td></td>
</tr>
<tr>
<td><strong>Stuart Zhu</strong></td>
<td><strong><a href="mailto:szhu28@illinois.edu">szhu28@illinois.edu</a></strong></td>
<td></td>
</tr>
</tbody>
</table>
Welcome to CS411 (N3, N4 & CPS)

- Web site is up at:
  - [https://wiki.illinois.edu/wiki/display/CS411SP19/Overview](https://wiki.illinois.edu/wiki/display/CS411SP19/Overview)
  Syllabus, policies, schedule, ...
  - Please read the class syllabus, policies, and schedule (tentative, will be updated constantly)

- Piazza signup link:
  - [piazza.com/illinois/spring2019/cs411](piazza.com/illinois/spring2019/cs411)
  - Please use your first and last name when you signup on Piazza

- Grades will be posted on Compass
Our class statistics:
By Class Standing
Our class statistics: By Level

Grad

Undergrad
Our class statistics: By Gender

![Bar graph showing class statistics by gender](image)
Our class statistics: By College

- School of Information Sciences
- Liberal Arts & Sciences
- Graduate College
- Engineering
- Education
- Division of General Studies
- Agr, Consumer, & Env Sciences

CS@Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Our class statistics:
By 1st Major
Course Agreement!

**Student Expectations**

a. PLEASE BE ACTIVE AND PARTICIPATE IN CLASS.
b. Listen and respect others.
c. Be comfortable taking risks.
d. Complete all assignments.
e. Turn off your cell phones.
f. Be punctual for all classes.
g. Discuss class concerns either after class or during designated office hours.
h. Be prepared for class by reading the assigned reading prior to lesson.
i. Make sure iClicker is with you, has batteries, and is working.

**Instructor Expectations**

a. BE ACTIVE AND ENTHUSIASTIC TO FACILITATE YOUR LEARNING.
b. Listen and respect your views.
c. Be in class at least 5 minutes before and after class.
d. Respond swiftly and effectively to your concerns.
e. Turn off cell phone.
f. Grade objectively, consistently, and in a timely manner.
g. Be prepared for class.
h. Accommodate differences in your learning.

Source: Constructing a Learner-Centered Syllabus: One Professor’s Journey
CS411: All about “Databases”

DBMS (Data Base Management System) = Database Systems = Databases

*System to manage, maintain, query, interact with, transact with data.*

More loosely, database systems are used for “data management”
CS411 Goals: Two Perspectives of DBMS

• USER PERSPECTIVE
  – how to use a database system?
  – conceptual data modeling, the relational and other data models, database schema design, relational algebra, SQL and No-SQL query languages.

• SYSTEMS PERSPECTIVE
  – how to design and implement a database system?
  – data representation, indexing, query optimization and processing, transaction processing, and concurrency control.
  – NOT COMPLETE: high-level view of implementation; CS511
Why is This Course in the Curriculum?

• It integrates CS concepts
  Languages, data structures, concurrency
• Most CS courses concentrate on code – our interest is managing and representing data and data-centric computation
• It teaches valued job skills
  – DB design and modeling, SQL, NoSQL, Web technologies.
• Important to learn data management
• Example of the practical power (query optimizers) of an underlying theory (relational algebra)
• Databases are now part of a larger data management ecosystem underneath the Web
Prerequisites

• Must have data structure and algorithms background
  – CS 225 or equivalent assumed

• Good programming skills
  – Project will require lots of programming
  – Need C++, Java, Python, or PHP … to communicate w/ the DB
  – Your project group picks the language
  – We cannot help with debugging for your language
    • (i.e., pick wisely)
Textbook

• Textbook:

Database Systems: The Complete Book, 2/e,
by Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer D. Widom
All readings will come from this book.
(Looks intimidating, but it’s not very dense...)

• Good references:
  – Database Management Systems,
    by Raghu Ramakrishnan and Johannes Gehrke,
    McGraw-Hill
  – Database System Concepts, by Abraham Silberschatz,
    Henry F. Korth, and S. Sudarshan, McGraw Hill
Course Format

• For all students
  – two 75-min lectures / week
  – 10 assignments planned (MPs/Paper-based)
  – project (significant)
    • Several stages
  – Two midterms (in class)
    • Cheat sheet allowed, possibly in both

• Graduate students: 4 credits option
  – write a survey paper
Lectures

• Lecture slides will be posted shortly before the lecture
  – Are meant to complement the textbook
• Lectures are important for guiding your reading of textbook (and will be covered in exams and homework assignments)
  – So please attend lectures or watch videos regularly
In-class activities

• This semester, we will be using iClickers for in-class participation.
• Each lecture, we will have 1-4 iClicker activities to encourage collaboration and engagement with the class material.
• You must answer questions throughout the class to get participation credit.
iClicker

- You can use Clicker 2 ("iClicker")
  - available at the Illini Union Bookstore
- Or you can download iClicker Reef Student App at
  - [https://www.iclicker.com/students/apps-and-remotes/apps](https://www.iclicker.com/students/apps-and-remotes/apps)
- To get participation credit, your iClicker must be registered on the Compass2g site for the course.
- **You are responsible for ensuring your iClicker is with you, has batteries, and is working.**
Homework Assignments

• 10 weekly paper-based and programming assignments
  – We’ll drop the lowest grade HW
• Assignments will be posted on the wiki page
• Submit through GradeScope
• **No** late homework will be accepted
  – Late = missing
  – OK to submit partial work

For more details, see policy page
Project

• Build a Web-based Database-driven Application
  – select an database application that needs a database
  – design and build it from start to finish
  – your choice of topic:
    useful, realistic, database-driven, web-based, fun!
• Easy strategy to pick projects: pick your favorite web company, re-implement a tiny version of it!
  – A tiny Amazon: managing and selling inventories of books
  – A tiny Facebook: keeping track of all your posts, likes, and friendships
  – A tiny AirBnB: matching home-owners to visitors

Hopefully you’ll take away the fact that all web-based companies (or all companies) require databases at their core.
Project

• Team work (3-4): **not lower!**
  – Would suggest starting at 4 rather than 3, given amount of work

• Significant amount of programming (we will provide 2 tutorials)

• Will be done in stages
  – you will submit some work at the end of each stage

• You will show a demo at the end of the semester
Project Groups

- Project will be done in group of 3-4 students
  - learn how to work in a group: valuable skills
  - may also use project group as study partners
  - fulfills your curricular requirements for project-based activities!

- Try to form groups as soon as possible
  - can start by posting requests on piazza
  - especially important for online students
Project Groups

• There will be a deadline soon for forming groups
  – if you have not formed groups by then
  – we will help assign you to groups

• Grading:
  – all members receive same grade
  – if someone drops out, the rest pick up the work (sorry!)
  – Another benefit to starting at 4, not 3
Four Credit Version

• Only for Graduate students
• Read and summarize the literature on a specific topic
  – From a selected list of topics
• Submit a survey report + presentation
Exams

• Two Midterms
  – Both in class.

• The schedule is up. Check exam dates and mark them on your calendar
  – you should not have conflicts if you are able to take the class
  – generally no makeup exams unless exceptional cases (see policy page)
Tentative Grading Breakdown

• Homework: 25%
• Project: 30%
• Participation 5%
  – On-campus students 2% Piazza + 3% in-class activities
  – Online students: 5% Piazza participation
• 1st Midterm: 20%
• 2nd Midterm: 20%

For 4 Credit students: 90% = the total of everything above, 10% = Survey Project
IMPORTANT: Plagiarism

• We have a zero-tolerance policy on plagiarism.

• We will be running automated plagiarism detection software on all your course submissions.
  – This has led to dozens of students being docked grades.
  – Do not do this!

• Similarly, do not attempt to copy in class
  – We will find you
IMPORTANT: Communications and Contacting the Staff
Communications: From us to you

Piazza:
- Signup link: piazza.com/illinois/spring2019/cs411

- vitally important!
- announcements will be posted on Piazza
- make sure to check it regularly for questions/clarifications
- Enable “notifications”
Communications: From You to Us

We’re a LARGE class (250ish) students.

If you have a question/problem

1. talk to people in your group first
2. post your question on piazza
3. if it is of a sensitive nature, post your question *privately* to piazza
4. if it is pertinent only to one TA, and of a sensitive nature, then email the TA
5. if it is sensitive & not easily conveyed electronically, then go to office hours to talk to TA or instructor
Piazza

• Designed for you and your peers
  – to communicate and help one another
  – please do not post solutions
• TAs will monitor relatively regularly and try their best to help with your questions
  – But don’t expect responses in <24 hours.
    • There will be many questions
    • May not be able to answer all of them in timely manner
    • Don’t wait until the last minute to ask!
  – Not good for more complex questions -- come to office hours or email TA
Piazza: Incentivizing Participation

• Since it’s hard for us to be present 24x7 on Piazza, we want to incentivize the students who diligently answer other’s questions

• Up to **2% of the grade** for on-campus students and **5%** for online students for Piazza participation
  – Grading criteria: informative, succinct, useful, clear answers are rewarded.
Office Hours

• For any complex in-person questions and clarifications

• Teaching staff has office hours six times a week, at least one each day of the week.
  – Mine: Friday 10:45-11:30 (SC 4209)

• See Piazza staff page for schedule:
  – https://piazza.com/illinois/spring2019/cs411/staff
IMPORTANT: Registration Questions

• I get 2-3 emails a day about registration/wait-list questions.
  – Unfortunately, I won’t be able to respond to all emails

• We will not maintain a waitlist

• Registration will be first-come-first-serve.

• For any questions regarding registration, please talk to the CS academic office.
IMPORTANT: Grading Policy

• In the past we have tried two forms of grading:
  – Absolute, score-based grading
    
    | Total   | Grade          |
    |---------|----------------|
    | 90-100  | A (A-, A, A+)  |
    | 80-89   | B (B-, B, B+)  |
    | 70-79   | C (C-, C, C+)  |
    | 60-69   | D (D-, D, D+)  |
  – And curving

• And then we take the best of both grades.
• Typically, curving leads to a higher grade.
• Separate curves for 4/3 credit, and ug/grad, but hasn’t ended up mattering in the past.
Onto databases now!

LET'S GET STARTED!
Task: Build a Banking System from Scratch

Goal: Manage customers, accounts, joint accounts, transfers, transactions, loans.

Let’s say I implement this system using C++ or Java, without using a database.

Think like a designer: what aspects do we need to worry about?
Aspects to worry about

- Deal with lots of data
- Be fast
- Don’t lose information
- Allow multiple users
- Stay consistent
- Easy to use
The Database Approach

• Abstract out all of the data management functionality into a separate layer
• Many applications can access it
• Turns out this “separate layer” keeps turning up in many many many scenarios
• Makes sense to abstract it out
Database Management System (DBMS)?

System for providing EFFICIENT, CONVENIENT, and SAFE, MULTI-USER storage of and access to MASSIVE amounts of PERSISTENT data
Example: Banking system

System for providing EFFICIENT, CONVENIENT, and SAFE, MULTI-USER storage of and access to MASSIVE amounts of PERSISTENT data

- Data = information on accounts, customers, balances, loans, transaction histories, etc.
- MASSIVE:
  - many TBs at a minimum for big banks, more if keep history of all transactions,
  - even more if keep images of checks -> Far too big for memory
Example: Banking system

System for providing **EFFICIENT, CONVENIENT, and SAFE, MULTI-USER** storage of and access to **MASSIVE** amounts of **PERSISTENT** data

- **PERSISTENT**: data outlives programs that operate on it, even on
  - system shutdown
  - power failure

So simply can’t store these things in memory, we have to rely on stable storage (disk, flash)
MULTI-USER Access

System for providing EFFICIENT, CONVENIENT, and SAFE, MULTI-USER storage of and access to MASSIVE amounts of PERSISTENT data

• MULTI-USER: many people/programs accessing same database, or even same data, simultaneously -> Need controls
• Alice @ ATM1: withdraw $100 from account #002
  get balance from database;
  if balance >= 100
    then balance := balance – 100; // dispense cash
  update balance in database;
• Bob @ ATM2: withdraw $50 from account #002
  get balance from database;
  if balance >= 50
    then balance := balance - 50; // dispense cash
  update balance in database;

Initial balance = 100. What’s the ideal case? What could go wrong?
DBMS: More Requirements

System for providing EFFICIENT, CONVENIENT, and SAFE, MULTI-USER storage of and access to MASSIVE amounts of PERSISTENT data

• SAFE:
  – from system failures. E.g., money should not disappear or appear from the account, due to a power failure!
    Bob @ ATM2: withdraw $50 from account #002
    get balance from database;
    if balance >= 50
      then balance := balance - 50; // dispense cash
    update balance in database;
  – from malicious users
DBMS: More Requirements

System for providing **EFFICIENT, CONVENIENT, and SAFE, MULTI-USER** storage of and access to **MASSIVE amounts of PERSISTENT** data

- **CONVENIENT:**
  - simple commands to debit account, get balance, write statement, transfer funds, etc.
  - also unpredicted queries should be easy
  - shouldn’t require complex 100s of lines of code

- **EFFICIENT:**
  - don't search all files in order to get balance of one account, get all accounts with low balances, get large transactions, etc.
Why Direct Implementation Won’t Work / is Super Hard

- Early DBMS evolved from file systems
- Provided storage of **MASSIVE** amounts of **PERSISTENT** data, to some extent
- **SAFE?**
  - when system crashes, no guarantees on how program may behave: we may lose data
- **EFFICIENT?**
  - Does not intrinsically support fast access to data whose location in file is not known: will need to write custom code
Why Direct Implementation Won’t Work

• CONVENIENT?
  – need to write a new C++/Java program for every new query
  – small changes to structure entails changing file formats; need to rewrite virtually all applications

• MULTI-USER ACCESS?
  – limited protection
  – need to worry about interfering with other users
That’s why the notion of DBMS was invented!
DBMS: A Software System

• Buy, install, set up for particular application or applications

• Major vendors:
  – Oracle
  – IBM (DB2)
  – Microsoft (SQL Server, Access)
  – Sybase

• Open source:
  – Postgres
  – MySQL
  – Sqlite

➤ All are "relational" DBMS
DBMS Examples

• Most familiar use: many Web sites rely heavily on DBMS's

• And many non-Web examples
DBMS Architecture

Pretty complex piece of software!!
Data Model and Schemas

• Defining the data model of the database:
  – conceptual structuring of data stored in database
  – E.g., data is set of records, each with student-ID, name, address, courses, photo
  – E.g., data is graph where nodes represent cities, edges represent airline routes

• Schema versus data
  – schema describes how data is to be structured, defined at set-up time, rarely changes (also called "metadata")
  – data is actual "instance" of database, changes rapidly
People / Roles

• DBMS application designer
  – set up schema, loads data, ...
  – Primarily DDL (Data Definition Language)

• DBMS user: queries/modified data
  – Primarily DML (Data Manipulation Language)

• DBMS administrator (DBA)
  – user management, performance tuning, ...

• DBMS implementer: builds systems
First ½ Topics: User Perspective

• SQL and DBMS Functionalities:
  – SQL Programming
  – Queries and Updates
  – Indexes and Views

• Entity-Relationship Model

• Relational Model

• Relational Database Design

• Relational Algebra
Second ½ Topics: System Perspective

- Storage and Representation
- Indexing
- Query Execution and Optimization
- Transaction Management
- NoSQL Databases:
  - Graph Databases (Neo4J)
  - Document-oriented Databases (MongoDB)
How to Get the Most out of CS411?

• Read and think before/after class
  – readings are there for a reason
  – discuss assignments w/ others but write your own solution!

• Use lectures as a guide
  – a roadmap for what’s important
  – lectures are starting points– they do not cover everything you should read
Questions?