THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

1. Course Overview

CSCI 2541 Database Systems & Team Projects

Wood & Chaufournier

Slides adapted from ones by Prof. Bhagi Narahari

Last time…

Structure that is **independent** of the underlying file formats

Queries to flexibly read, update, and delete information

Transactions that provide guarantees about **multi-user consistency**

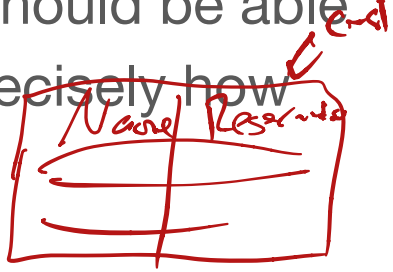
…this time.

Queries & Data Independence

Queries to flexibly read, update, and delete information

What this means….

A user of a relational database system should be able to use the database without knowing precisely how data is stored, e.g.




```
SELECT Name, Reservation
FROM Customers
WHERE Name= 'Billy Miller'
```

The above "query" does not need to know how the data in Customers is stored. Why should you need to worry about that?!

How to define and use a database

Queries to flexibly read, update, and delete information

Data **Definition** Language (**DDL**) to specify database schema

- What data, and how it is organized (**logical level**)

Data **Manipulation** Language (**DML**) allows users to access or manipulate data as organized by data model

- procedural DMLs: require user to specify **what data and how** to get it
- non-procedural DMLs: require user to specify what data is needed **without** specifying how to get it.

Often, one language provides both features (e.g., SQL)

Relational DB Query Languages

Queries to flexibly read, update, and delete information

Formal query languages:

- **Relational algebra,**
- Relational Calculus,
- Why study formal languages?

Commercial query language: **SQL**

SQL: "descendent" of SEQUEL; mostly relational algebra and some aspects of relational calculus

- has procedural and non-procedural aspects
- Has DDL and DML components

What is SQL?

Queries to flexibly read, update, and delete information

SQL is **not** a specific database software

It is a standardized query language

- Defines how to create a database schema and issue read/write queries

DBMS software must implement the SQL **standard**

- Plus some of their own extensions
- None actually follow the official ANSI SQL standard precisely…

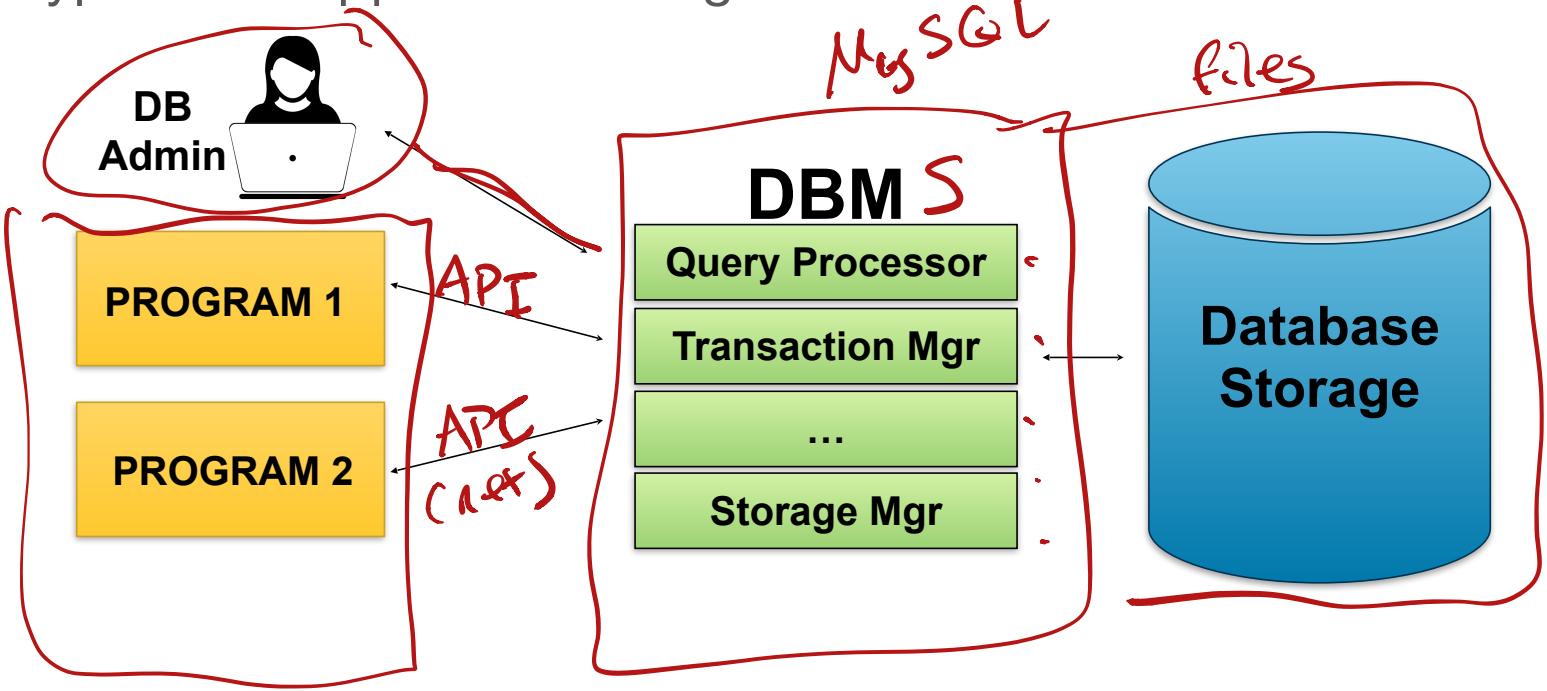
Good news: SQL queries are "cross platform" and will work on many different database systems

Confusing? Curious?

Connecting to a DB

Transactions that provide **multi-user consistency**

A typical DB Application design



DBMS

Transactions that provide **multi-user consistency**

The data abstraction is provided by the DBMS

- Separation b/w Logical and Physical, Query language parsing, multi-user, etc.

A database management system provides efficient, convenient, and safe multi-user storage and access to massive amounts of persistent data

- **Efficient & Convenient** - Able to handle large data sets, complex queries without searching all files and data items, easy to write queries
- **Scalability** – Large/huge data
- **Persistence & Safety** - Data exists after program execution completes, handles loss of power
- **Multi-user** - More than one user can access and update data at the same time while preserving consistency….concept of transactions

Components of a DBMS

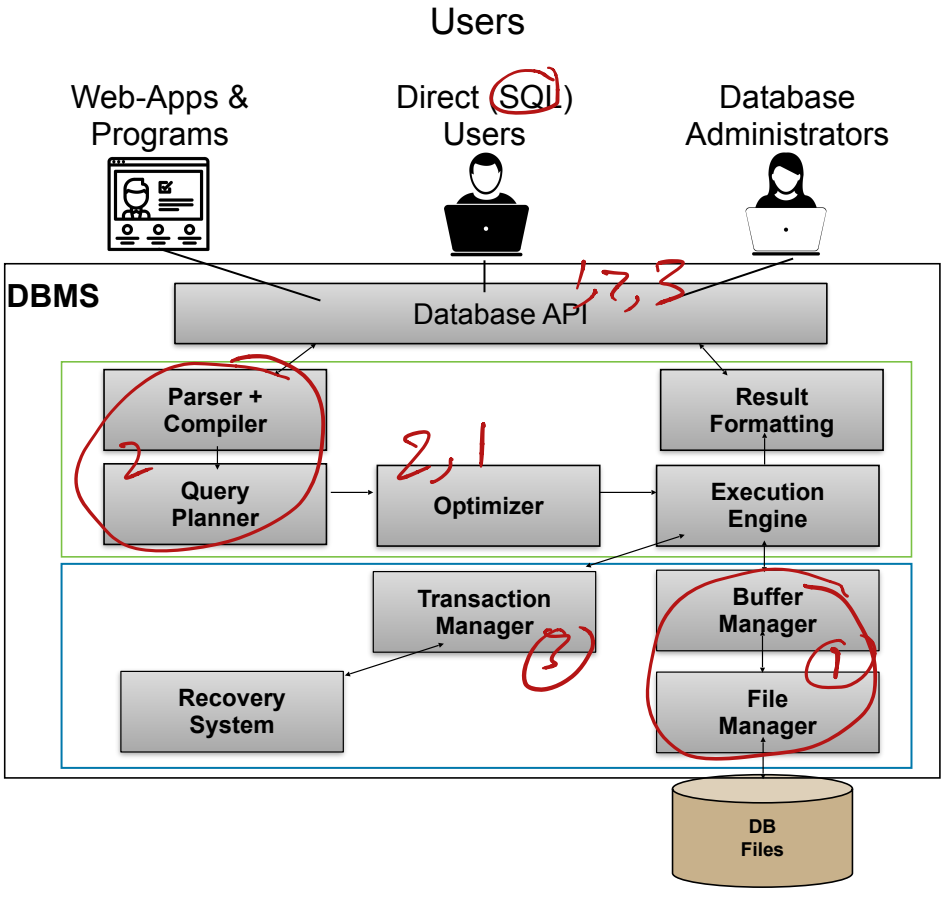
Transactions that provide **multi-user consistency**

A database management system provides efficient, convenient, and safe multi-user storage and access to massive amounts of persistent data.

A DBMS is a complicated software system containing many components:

- **Query processor** - translates user/application queries into low-level data manipulations
  - Sub-components: query parser, query optimizer
- **Storage manager** - maintains storage information including memory allocation, buffer management, and file storage
  - Sub-components: buffer manager, file manager
- **Transaction manager** - performs scheduling of operations and implements concurrency control algorithms
  - You will learn more about storage management and concurrency in the Operating Systems course… enjoy!

DBMS Architecture: Complete Picture



Structure that is **independent** of the underlying file formats

Queries to flexibly read, update, and delete information

Transactions that provide **multi-user consistency**

Which components provide these properties?

This course is about Database Design…

Focus is on design of databases

- Working at the **logical level**

Internals of DBMS is not the focus in this course

- BUT we will touch upon a few key concepts that make DBMS' work

- DBMS design brings together several key concepts from Computer Science
  - Languages, Compilers/translation, Algorithms, Data structures, Operating systems…

Database System Architectures

There are several different database architectures:

- **Embedded architecture** - DB files and DBMS processing occurs at the clients (e.g. Microsoft Access or SQLite)
  - You will probably work with this in Systems Programming 3410

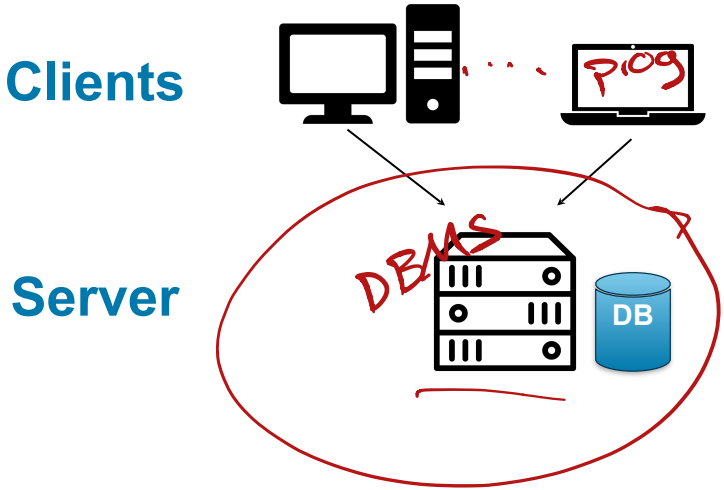
2 tier

- **DB client-server architecture** - dedicated machine running DBMS accessed by remote clients (e.g. MS SQL Server, MySQL, Postgres)

- **Multi-Tier client-server architecture** - DBMS is bottom tier, second tier is an application server containing business logic, top tier is clients (e.g. Web browser-Apache/Tomcat-Oracle)
  - i.e., a LAMP Stack

DB Client-Server Architecture

2 tier



Tier 1: Client Application
- User Interface
- Business and Data Logic

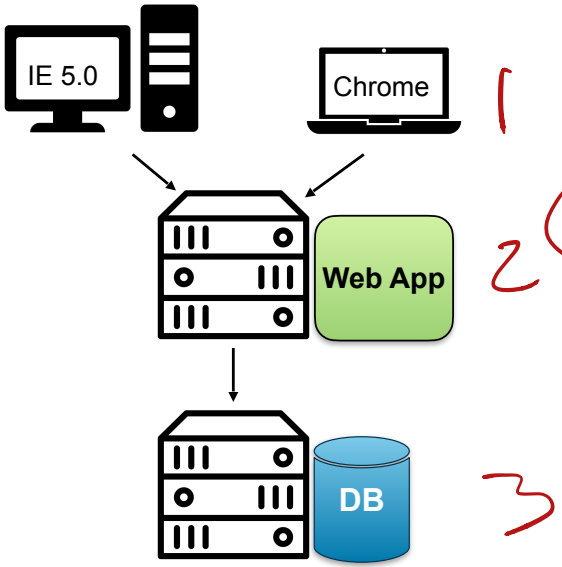
Tier 2: Database Server
- Server-side validation
- Data storage/management

DBMS

Advantages:

- Only one copy of DBMS software on dedicated machine
- Increased performance
- Easier to maintain consistency and manage concurrency

Three-Tier Client-Server Architecture – our approach



Clients

Browser

Form

"23 jews"

1

2

3

Tier 1: Client (Web/mobile)
- User Interface

HTML/CSS/Javascript

Tier 2: Application Server
- Business logic & data processing

Python+Flask

Tier 3: Database Server
- DBMS and data storage

MySQL

Advantages:

- Reduced client administration and cost using thin web clients.
- Easy to scale architecture and perform load balancing.

Course Requirements & Logistics…

Course Objectives

Relational database theory and design

- Concepts of data storage and retrieval

Fluency in SQL and web front- and back-end development

- Working with relational database systems: MySQL
- Python to develop DB connected apps

Software integration experience and team S/W development experience

- Design and deploy a large database application
- Full stack (web) development

Brief introduction to NoSQL database models

Course Schedule - Topics

Part 1: Relational Databases. Weeks 1-6

- Relational model & Formal query languages (Rel. Algebra)
- SQL – query language, and MySQL DBMS
- Python (and brief review of HTML/CSS – webpage design)
- Relational Schema Design
  - Entity-Relationship (ER) Model
  - Normal forms and DB tuning
- Overview of DBMS: Security, File manager/Indexing

Part 2: Project (Teams). Weeks 7-14

- Full stack development, Integration of modules, Team S/W Dev

Part 3: Intro to Databases (& Analytics) for Semi/Un-structured Data. Weeks 10-12

- NoSQL DB Models; Experience working with MongoDB

Writing requirements (WID) – CS2501& final project report

In-class work

You will learn through in-class activities/demos and exercises most classes (lecture+lab)

- Must read the material and come to class → Slack

If you are assigned an exercise during class (i.e., an "in class exercise"), you need to complete the exercises within 24 hours - no exceptions!

- We will use Zoom breakout rooms for group work
- We may ask a group to present solutions to class

Async Attendance

We prefer you attend lectures/labs “live”

- But depending on your time zone this may not be feasible

Asynchronously watching the videos and completing the lab assignments is also acceptable

You still need to find a way to participate!

- This will be harder since you can’t “raise hand”
- Post questions to slack under each class’s thread
- Look for bonus activities in #engage

Course Resources

Course webpage: will have links to syllabus, lecture notes, online resources (and inclass exercises when applicable)

- cs2113f18.github.io

Github:

- Used to post and submit 'lab' assignments (requiring code)
- Project submissions
- Team project management

Blackboard:

- Electronic submission of non-programming homeworks
- Reporting grades

Slack:

- For class announcements and Q&A
- For team coordination

Course Requirements: Grading

Exam (midterm): 22.5%

- Based on lectures and labs. Around Week 6-7

Homework, Lab Assignments, In-class: 25%

- Programming and written homework
- In-Lab/Class exercises given out during class and equivalent to a "quiz"

Team Project (and Teamwork): 47.5%

- Phase 1 (15%) + Teamwork (10%) + Phase 2 (22.5%)
- No final exam BUT final project demos are required
- To pass project, your demos have to work…NO partial credit.

Engagement: 5%

- Participation in class and online (at least once per week)
- A variety of bonus opportunities

Grades curved (and scaled as percentage of highest score in class)

Approximate grading method after curving and scaling

A- to A: 90-100%
B- to B+ : 80 to < 90
C- to C+: 70 to < 80
D-: >60

The Project

A significant part of your grade for the course is a large database systems project

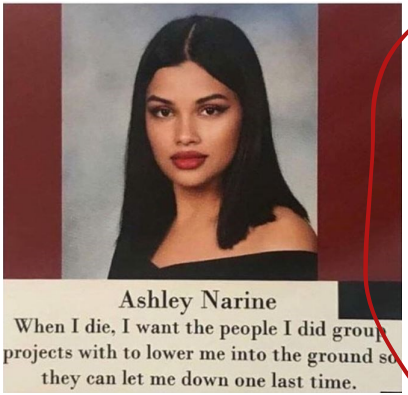
In the project you will design & implement a database system

- Full stack development:
  - Front End (HTML/CSS & optional Javascript)
  - Application server – in Python
  - DBMS backend – MySQL
- All the above are useful (high demand) skills
  - Note that limited background will be given on web programming (seek help from TAs)

The project will involve working in teams of 2 to 4

- Larger teams must develop projects with more features

Why do we have team projects



Real World: Teamwork and S/W development in teams is the default!

- Communication
- Collaboration
- Conflict resolution
- Using tools to enable collaborative SW dev

Team Project: Requirements & Expectations

Project broken into 2 (+ 0.1) phases:

- **Phase 0.1**: theoretical (paper) design of the database (ER model)
- **Phase 1**: teams to build an application assigned to your team
- **Phase 2**: Work in new teams to integrate different applications and produce the final project
  - Different teams may be assigned different projects
  - This requires **integration** and **NOT redesign**
  - Take what you built in Phase 1 and integrate with systems built by others….
    - If you "hide" in Phase 1, then you will be exposed in Phase 2 !!

You HAVE to deliver a working project…else Zero on project

Agile SW Development process

- Build the system iteratively rather than all in one (giant) step
- works well with your teamwork assessment (weekly check-ins)

Teamwork Assessment…part of your grade!

You have to work in teams

- Each team member required to ‘produce’ equitable share ‘product’
- Teamwork will be assessed…
  - Not all team members may get the same grade on the project!
  - You must bring teamwork issues to attention of the instructor

The second half of the course will have one session (lecture or a lab) dedicated to teamwork check-ins each week

- Instruction team will meet with each team, and assess if the weekly deliverables are being met by each team member

If you cannot commit time each week to working on the team project then please drop the course!

- If you do not want to work in a team and do the work, then we do not want your attitude to negatively impact other students

Lab Sections: treated as one lab section

Lab sections conducted by the instructor(s) and TAs:

- Lead Instructor for Labs: Lucas Chaufournier

Lab sections will cover:

- HTML/CSS
  - Javascript – tutorials provided by TAs/UTAs during office hours
- MySQL
- Python
- Intro to a NoSQL DB - MongoDB
- Clarifications on Programming Assignments

In-class assignments in some weeks

- Will be graded, with async option

Academic Integrity Policy

No collaboration (of any sort) on homeworks/ programming assignments

- Including external resources, tutors, online
- Okay to clarify questions
- Not Okay to share solutions
- Not okay under any circumstances to share or show Code

No collaboration between teams on team projects

- within team each team member must have clear role -- i.e., clearly partitioned tasks for each team member

Academic Integrity

Strictly enforced! You are here to learn – so keep that in mind

Today's CS job process: Technical interview is the first step – **employers do not care about your 4.0 GPA if you do not pass the first technical interview!**

- You need the skills taught in this course! Seek help if you are behind!

Violations will lead to at least a zero on the work and a grade lower than final grade

Stay on top of your work – and come ask us questions!

PDT: Plagiarism detection software tool

- We may be running code submissions through software tool
- Any pair of submissions with more than 25% similarity will be closely examined

HTML/CSS Time!

Attributions

These slides are adapted from materials made by Prof. Bhagi Narahari

Image attribution: