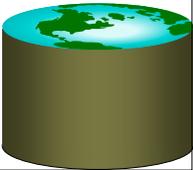


# CS186: Introduction to Database Systems

Philip Bohannon  
Brian Cooper

Spring 2008

Much of the nice slideware is courtesy Joe Hellerstein



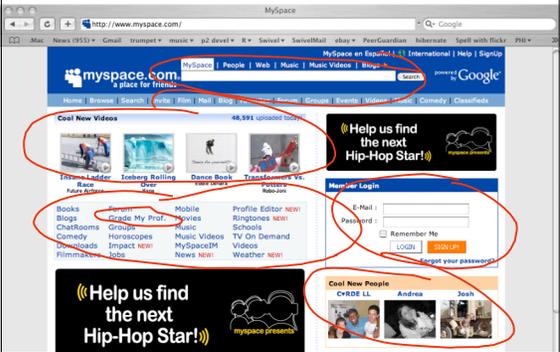
## This course (a tour by interrogative pronoun)

- What?
- Why?
- Who?
- How?
- For instance?

## What: Database Systems Then



## What: Database Systems Today



## What: Database Systems Today



## What: Database Systems Today

Account	Account Number	Available Balance
CHECKING	111-200800	\$7,289.48
SAVINGS	557-291100	\$189.46
SAVINGS	111-153500	\$1,262.65
<b>Total</b>		<b>\$8,738.57</b>

Account	Account Number	Total Account Value
BROKERAGE	W074000000	\$15,866.56
<b>Total</b>		<b>\$15,866.56</b>

Account	Account Number	Outstanding Balance	Available Credit
MASTERCARD	5490-9600-0000-3000	\$1,631.79	\$6,668.21
<b>Total</b>		<b>\$1,631.79</b>	<b>\$6,668.21</b>

Account	Account Number	Outstanding Principle Balance
STUDENT LOAN	70004000	\$5,000.00
<b>Total</b>		<b>\$5,000.00</b>

## What: Database Systems Today

## What: Database Systems Today

## So... What is a Database?

- We will be broad in our interpretation
- A Database:
  - collection of interrelated data + description of data
- A Conceptual Model to Describe Data
  - Entities (e.g., teams, games)
  - Relationships (e.g. *The A's are playing in the World Series*)
- Might surprise you how flexible this is
  - Web search:
    - Entities: words, documents
    - Relationships: *word in document, document links to document.*
  - P2P filesharing:
    - Entities: words, filenames, hosts
    - Relationships: *word in filename, file available at host*

## What is a Database Management System?

- A Database Management System (DBMS) is:
  - A software system designed to store, manage, and facilitate access to databases.
- Typically this term used narrowly
  - Relational databases with transactions
    - E.g. Oracle, DB2, SQL Server
  - Mostly because they predate other large repositories
    - Also because of technical richness
  - When we say **DBMS** in this class we will usually follow this convention
    - But keep an open mind about applying the ideas!

## Database Systems and the World of Data

- Database systems think they are the center of the world!
  - Who can blame them - often they are!
- But our definition says nothing about the software, and...
- Powerful computers + bandwidth means easy-to-get databases
- Even in enterprise, database systems appear...
  - in dozens of places
  - playing dozens of roles
  - part of complex data flows

## What: Is the WWW a DBMS?

- That's a complicated question!
- The "surface web": docs and search
  - Crawler *indexes* pages on the web
  - Keyword-based *search* for pages
- Web-cache SW at Google/Yahoo is a kind of DBMS
- Notes
  - source data is mostly "prose": *unstructured* and *untyped*
  - public interface is *search only*:
    - can't modify the data
    - can't get summaries, complex combinations of data
  - few guarantees* provided for freshness of data, consistency across data items, fault tolerance, ...



## What: "Search" vs. Query

- Try **actors who donated to presidential candidates** in your favorite search engine.
- Now try **engineers who donated to presidential candidates**



- If it isn't "published", it can't be searched!



## What: A "Database Query" Approach

Actors in the Yahoo! Directory

- Acting@
- Book@
- Fan@
- Organization
- Resumes@
- Web Direct

Presented by the Federal Election Commission

Individuals Who Gave To: **KERRY, JOHN F**

Sorted By Transaction Type Then Last Name

Committee(s) Used In This Query:

- KERRY-EDWARDS 2004 INC.
- KERRY-EDWARDS 2004 INC. GENERAL ELECTION LEGAL AND ACCOUNTING COMPLIANCE FUND
- JOHN KERRY FOR PRESIDENT, INC
- VETERANS FOR JUSTICE

The query you have chosen matched 222599 individual contributions.



## "Yahoo Actors" JOIN "FECInfo" (Courtesy of the Telegraph research group @Berkeley)

Query Finished Yahoo FECInfo (www.tray.com)

Results

Q: Did it Work?

Name	Occupation	Address	Amount
Smits, Jimmy	Self employed	Los Angeles, ...	250.00
Somers, Suzanne	Self	Valencia, CA ...	1,000.00
Stamp, Terence	Info Requested	Sanbornville, ...	1,000.00
Stone, Sharon	Self employed/Actress	Los Angeles, ...	1,000.00
Streisand, Barbara	Self employed/Singer / Prod.	Santa Monica, ...	1,000.00
Taylor, Elizabeth	Not employed/Homemaker	Tampa, FL 33...	250.00
Thomas, Heather	CIGNA Healthcare/New Busi...	Nashville, TN ...	250.00
Thomas, Michelle		Washington, ...	300.00
Thomas, Olive	National Council of Church...	Maryville, TN ...	1,000.00
Thomas, Olive	National Council of Church...	Maryville, TN ...	1,000.00
Tomlin, Lily	Self employed/Actress	Los Angeles, ...	250.00
Triggleson, Jeanne	Self employed/Actress	Los Angeles, ...	1,000.00
Wagner, Robert	Self employed/Doctor	McLean, VA 2...	500.00



## What: Is a File System a DBMS?

- Thought Experiment 1:
  - You and your project partner are editing the same file.
  - You both save it at the same time.
  - Whose changes survive?

A) Yours B) Partner's C) Both D) Neither E) ???

- Thought Experiment 2:
  - You're updating a file.
  - The power goes out.
  - Which changes survive?

A) All B) None C) All Since Last Save D) ???



## What: Is a File System a DBMS?

- Thought Experiment 1:

Q: How do you write programs over a subsystem when it promises you only "???" ?

A: Very, very carefully!!

-Which changes survive?

A) All B) None C) All Since Last Save D) ???



## OS Support for Data Management

- Data can be stored in RAM
  - this is what every programming language offers!
  - RAM is fast, and random access
  - Isn't this heaven?
- Every OS includes a File System
  - manages *files* on a magnetic disk
  - allows *open, read, seek, close* on a file
  - allows protections to be set on a file
  - drawbacks relative to RAM?



## Database Management Systems

- What more could we want than a file system?
  - Simple, efficient *ad hoc*<sup>1</sup> queries
  - concurrency control
  - recovery
  - benefits of good data modeling
- S.M.O.P.<sup>2</sup>? Not really...
  - as we'll see this semester
  - in fact, the OS often gets in the way!

<sup>1</sup>*ad hoc*: formed or used for specific or immediate problems or needs

<sup>2</sup>SMOP: Small Matter Of Programming



## Current Commercial Outlook

- Relational DBs a major part of the software industry
  - Oracle, IBM, Microsoft, HP, Teradata, Sybase, ...
- Open Source coming on strong
  - Relational: MySQL, PostgreSQL, Apache Derby, SQLite, Ingres, ...
  - text-Search: Lucene, Ferret, ...
- Well-known benchmarks (TPC, TREC)
- Tons of applications, related industries
  - Alphabet soup!
- Related database technologies have niches
  - P2P, XML repositories, etc.



## What systems will we cover?

- We will be try to be broad and touch upon
  - Relational **DBMS** (e.g. Oracle, SQL Server, DB2, Postgres)
  - Larger world of **Data Management** (Extract, Transform & Load, Parallel Data Processing)
- Ground things in relevant applications



## Quiz Questions

- Is there any data you care enough about to manage?
- What if you start a company?
- Do your favorite apps require a database system?
- How can you tell?



## Why take this class?

- Database systems are at the core of CS
- They are incredibly important to society
- The topic is intellectually rich
- A capstone course for undergrad
- It isn't that much work
- Looks good on your resume
- Be a data ninja

Let's spend a little time on each of these



## Why take this class?

### A. Database systems are the core of CS

- Shift from computation to information
  - True in corporate computing for years
  - Web made this clear for "the rest of us" by the end of 90's
  - Increasingly true of scientific computing
- Need for DB technology has exploded in the last years
  - **Corporate**: retail swipe/clickstreams, "customer relationship mgmt", "supply chain mgmt", "data warehouses", etc.
  - **Web**: **not** just "documents". Search engines, maps, e-commerce, blogs, wikis, social networks. Web 2.0.
  - **Scientific**: digital libraries, genomics, satellite imagery, physical sensors, simulation data
  - **Personal**: Music, photo, & video libraries. Email archives. File contents ("desktop search").



## Why take this class?

### B. DBs are incredibly important to society

- "Knowledge is power." -- Sir Francis Bacon
- "With great power comes great responsibility." -- Spiderman's Uncle Ben



- Policy-makers should understand technological possibilities.
- Informed Technologists needed in public discourse on usage.



## Why take this class?

### C. The topic is intellectually rich.

- representing information
  - data modeling
- languages and systems for querying data
  - complex queries & query semantics\*
  - over massive data sets
- concurrency control for data manipulation
  - controlling concurrent access
  - ensuring transactional semantics
- reliable data storage
  - maintain data semantics even if you pull the plug

\* semantics: the meaning or relationship of meanings of a sign or set of signs



## Why take this class?

### D. The course is a capstone.

- We will see
  - Algorithms and cost analyses
  - System architecture and implementation
  - Resource management and scheduling
  - Language design, semantics and optimization
  - AI topics including logic and planning
  - Statistical modeling of data



## Why take this class?

### E. It isn't that much work.

- Bad news: It is a fair bit of work.
  - varies from year to year
  - this is more of an 'ebb' than a 'flow' term
- Good news: the course is front loaded
  - Most of the hard work is in the first half of the semester
  - Load balanced with most other classes



## Why take this class?

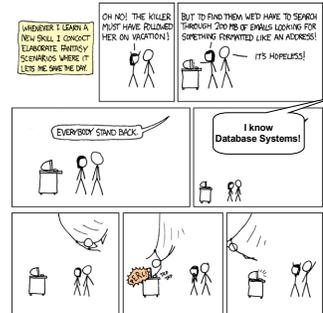
### F. Looks good on my resume.

- Yes, but why?
- Data Management is simultaneously the most boring and most interesting technology around!
- Database systems are "merely" a means to an end.
- We want cool applications.
- ...how long to prototype/build your new application?
- ...how long to add features?
- ...what happens when the power goes out, disk crashes, etc? (cool applications don't lose user data)



## Why take this class?

### F. Be a data ninja.





## Who?

- Instructors
  - Philip Bohannon
  - Brian Cooper
  - [BerkeleyCS186Spring2008Profs@yahoogroups.com](mailto:BerkeleyCS186Spring2008Profs@yahoogroups.com)
- TAs
  - Arsalan Tavakoli
  - Kenghao Chang



## How? Workload

- Projects with a “real world” focus:
  - Extract, Transform & Load real data from the web
  - Build a web-based application
  - Sort files on the grid with map-reduce
  - Modify the internals of a “real” open-source database system: PostgreSQL
- Other homework assignments and/or quizzes
- Exams – 2-3 Midterms & 1 Final
- Projects to be done in groups of 2
  - Pick your partner ASAP
- The course is “front-loaded”
  - most of the hard work is in the first half



## Instructors



- Team taught -
  - First 1/2, Philip Bohannon
  - Second 1/2, Brian Cooper
  - ...both on loan from Community Systems Group, Yahoo! Research.
- Teaching assistant team -
  - Keng-Hao Chang
  - Arsalan Tavakoli



## How? Administrivia

- <http://inst.eecs.berkeley.edu/~cs186>
- Office Hours:
  - PLB:
    - Tues 11:30-12:30
  - KC: TBA
  - AT: TBA
- Discussion Sections start next Monday



## How? Administrivia, cont.

- textbook
  - *Database Management Systems, 3<sup>rd</sup> Edition*
    - Ramakrishnan and Gehrke
  - *Agile Web Development with Rails, 2<sup>nd</sup> edition*
    - e-book is fine (better?)
  - *Programming Ruby, 2<sup>nd</sup> edition*
    - Free online
- Grading, hand-in policies, etc. will be on Web Page
- Cheating policy: zero tolerance
  - We have the technology...



## How? Administrivia, cont.

- Team Projects
  - Most will be teams of 2
  - Think about this now!
- Class Yahoo Group:
  - <http://groups.yahoo.com/group/BerkeleyCS186Spring2008>
    - read it regularly and post questions/comments.
    - mail broadcast to all TAs will not be answered
    - mail to the cs186 course account will not be answered



## Agenda for the rest of today

- A "free tasting" of central concepts in DB field:
  - queries and search
  - ➔ - data independence
  - transactions
- Next Time
  - the Relational data model
  - some XML tools
- Today's lecture is from Chapter 1 in R&G
- Read Chapter 2 and 27.6-27.7 for next class.



## Describing Data: 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 (fields, attributes) and keys.



## Example: University Database

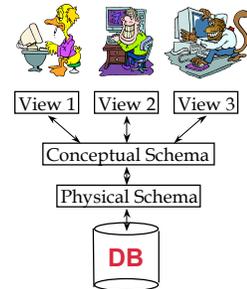
- Schema:
  - Students(sid text, name text, login text, age integer, gpa float)
  - Courses(cid text, cname text, credits integer)
  - Enrolled(sid text, cid text, grade text)



## Levels of Abstraction

Users

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used.



## Example: University Database

- Conceptual schema:
  - Students(sid text, name text, login text, age integer, gpa float)
  - Courses(cid text, cname text, credits integer)
  - Enrolled(sid text, cid text, grade text)
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - Course\_info(cid text, enrollment integer)



## Data Independence

- Applications (should be) 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.
- Q: Why is this particularly important for DBMS?

Because databases and their associated applications persist.



## Hellerstein's Inequality

$$\frac{dapp}{dt} \ll \frac{denv}{dt}$$



## Agenda ...

- A "free tasting" of central concepts in DB field:
  - queries (vs. search)
  - data independence
  - ➔ – transactions



## Concurrent execution of user programs

- Why?
  - Utilize CPU while waiting for disk I/O
    - (database programs make heavy use of disk)
  - Avoid short programs waiting behind long ones
    - e.g. ATM withdrawal while bank manager sums balance across all accounts



## Concurrent execution

- Interleaving actions of different programs: trouble!
  - Example:**
    - Bill transfers \$100 from savings to checking  
*Savings -= 100; Checking += 100*
    - Meanwhile, Bill's wife requests account info.
  - Bad interleaving:**
    - Savings -= 100
    - Print balances
    - Checking += 100
    - Printout is missing \$100 !



## Concurrency Control

- DBMS ensures such problems don't arise
- Users can pretend they are using a single-user system. (called "Isolation")
  - Thank goodness!



## Key concept: Transaction

- an **sequence** of database actions (reads/writes) executed **atomically** by DBMS
- should take DB from one **consistent state** to another



## Example



- Here, *consistency* is based on our knowledge of banking “semantics”
- In general, up to writer of transaction to ensure transaction preserves consistency
- DBMS provides (limited) automatic enforcement, via *integrity constraints*
  - e.g., balances must be  $\geq 0$

## Concurrent transactions

- Goal: **execute xacts  $\{T_1, T_2, \dots, T_n\}$ , and ensure a consistent outcome**
- *One option*: “serial” schedule (one after another)
- *Better*: **allow interleaving of xact actions, as long as outcome is equivalent to some serial schedule**

## Possible Enforcement Methods

- Optimistic: **permit arbitrary interleaving, then check equivalence to serial sched.**
- Pessimistic: **xacts set locks on data objects, such that illegal interleaving is impossible**

## Locking example

- T1 (Bill): *Savings -= 100; Checking += 100*
  - T2 (Bill's wife): *Print(Checking); Print(Savings)*
- T1 and T2 both lock Savings and Checking objects
- If T1 locks Savings & Checking first, T2 must wait

## A wrinkle ...

- T1 (Bill): *Savings -= 100; Checking += 100*
- T2 (Bill's wife): *Print(Checking); Print(Savings)*

Suppose:

1. T1 locks Savings
  2. T2 locks Checking
- *Now neither transaction can proceed!*
- called “deadlock”
- DBMS will **abort and restart** one of T1 and T2
- Need “undo” mechanism that preserves consistency
- Undo mechanism also necessary if system **crashes** between “Savings -= 100” and “Checking += 100” ...

## Ensuring Transaction Properties

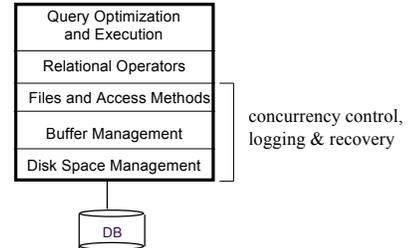
- DBMS ensures:
  - **atomicity** even if xact aborted (due to deadlock, system crash, ...)
  - **durability** of committed xacts, even if system crashes.
- *Idea*: Keep a *log* of all actions carried out by the DBMS:
  - Record all DB modifications in log, *before* they are executed
  - To abort a xact, undo logged actions in reverse order
  - If system crashes, must:
    - 1) **undo** partially executed xacts (ensures **atomicity**)
    - 2) **redo** committed xacts (ensures **durability**)
- *trickier than it sounds!*



## Architecture of a DBMS ...

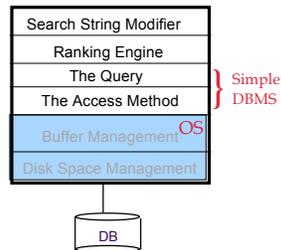


## Typical DBMS architecture



## A text search engine

- Less "system" than DBMS
  - Uses OS files for storage
  - Just one access method
  - One hardwired query
    - regardless of search string
- Typically no concurrency or recovery management
  - Read-mostly
  - Batch-loaded, periodically
  - No updates to recover
  - OS a reasonable choice
- Smarts: text tricks
  - Search string modifier (e.g. "stemming" and synonyms)
  - Ranking Engine (sorting the output, e.g. by word or document popularity)
  - Vague semantics: WYGIWIGY



## Advantages of a Traditional DBMS

- Data independence
- Efficient data access
- Data integrity & security
- Data administration
- Concurrent access, crash recovery
- Reduced application development time
- So why not use them always?
  - Expensive/complicated to set up & maintain
  - This cost & complexity must be offset by need
  - General-purpose, not suited for special-purpose tasks (e.g. text search!)



## Databases make these folks happy ...

- Web & enterprise app developers
- Computing infrastructure providers
- DBMS vendors, programmers
  - Oracle, IBM, MS ...
- End users in *many* fields
  - Business, education, science, ...
- Database administrators (DBAs)



...must understand how a DBMS works



## Summary

- Relational DBMS: maintain/query structured data
  - broadly applicable
  - can manipulate data and exploit *semantics*
  - recovery from system crashes
  - concurrent access
  - robust application development and *evolution*
  - data integrity and security



## Summary, cont

- Levels of abstraction & data independence.
- Goals of the course
  - 1) How to be a sophisticated user of database technology
  - 2) What goes on inside a DBMS and search engine
  - 3) How to architect data-intensive systems