CS186: Introduction to Database Systems

Joe Hellerstein

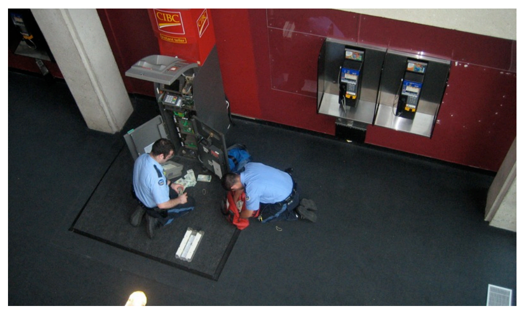
Spring 2009

Queries for Today

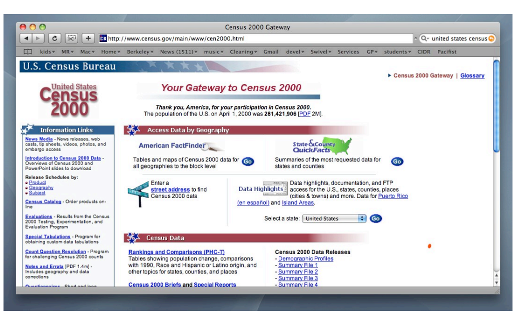
- What?
- Why?
- Who?
- How?
- For instance?

What: Spot the Database

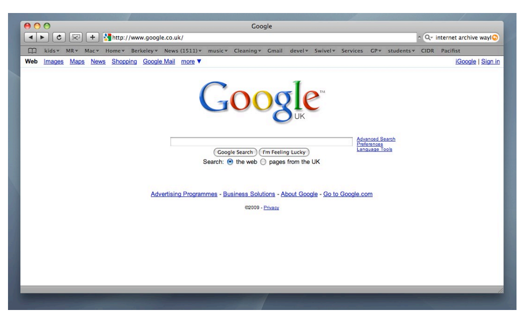
What: Spot the Database



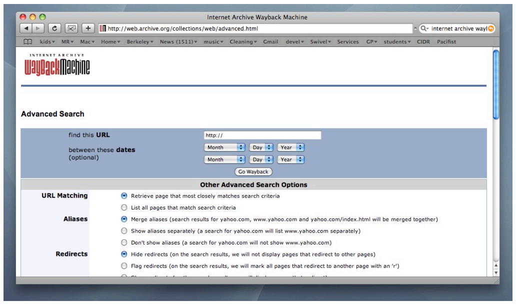
What: Spot the Database



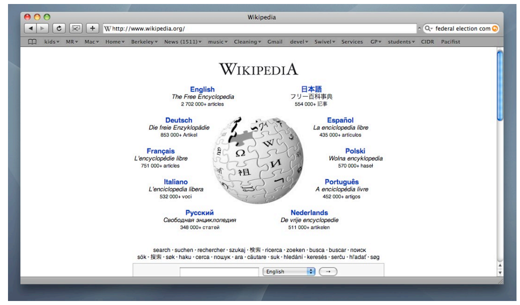
What: Spot the Database



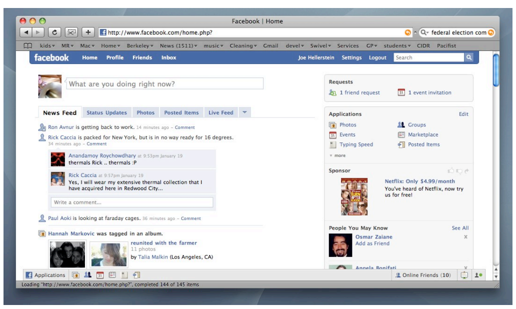
What: Spot the Database



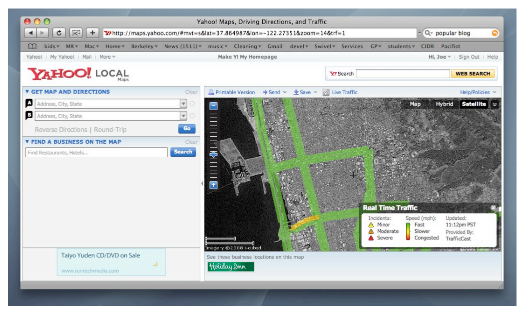
What: Spot the Database



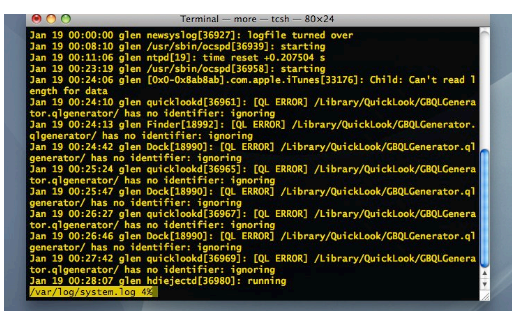
What: Spot the Database



What: Spot the Database



What: Spot the Database



So… What is a Database?

- We will be broad in our interpretation
- A Database:
  - A very large, integrated collection of data.
- Typically models a real-world "enterprise"
  - Entities (e.g., teams, games)
  - Relationships (e.g. The Raiders are not in the Superbowl)
- 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?

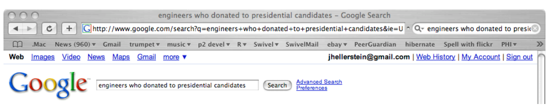
- A Database Management System (DBMS) is:
  - A software system designed to store, manage, and facilitate access to databases.
- Typically this term used narrowly
  - Relational databases with transactions
    - E.g. Oracle, DB2, SQL Server
  - Mostly for historical reasons
    - Also because of technical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

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

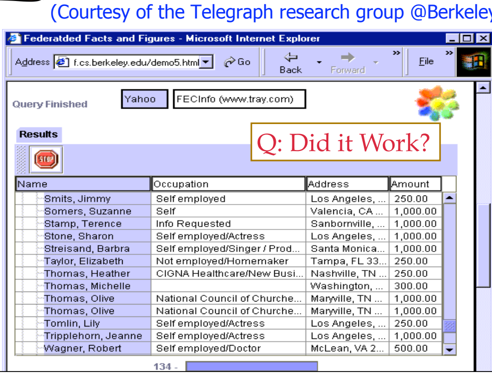


- If it isn't "published", it can't be searched!
  - But imagine a crawler that "auto-publishes" things

What: A "Database Query" Approach

"Yahoo Actors" JOIN "FECInfo"

(Courtesy of the Telegraph research group @Berkeley)



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, Barbra	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 Churche...	Maryville, TN ...	1,000.00
Thomas, Olive	National Council of Churche...	Maryville, TN ...	1,000.00
Tomlin, Lily	Self employed/Actress	Los Angeles, ...	250.00
Tripplehorn, 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:

ame

A) Y

E) ???

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*[1] queries
  - concurrency control
  - recovery
  - benefits of good data modeling
- S.M.O.P.[2]? Not really…
  - as we'll see this semester
  - in fact, the OS often gets in the way!

[1]ad hoc: formed or used for specific or immediate problems or needs
[2]SMOP: Small Matter Of Programming

Current Commercial Outlook

- Relational DBs a major part of the software industry
  - Elephants: Oracle, IBM, Microsoft, Teradata, Sybase, …
  - Startups: Greenplum, Aster, Cloudera, ParAccel, Vertica, …
- Obviously, Search
  - Google, Yahoo, MSN, Ask, …
- Public data services
  - Freebase, Many-Eyes, Swivel, DabbleDB
- Open Source coming on strong
  - Relational: MySQL, PostgreSQL, SQLite, Ingres, …
  - Text: Lucene, Hadoop
- Well-known benchmarks (TPC, TREC)
- Tons of applications, related industries
  - Alphabet soup!

What systems will we cover?

- We will be try to be broad and touch upon
  - Relational DBMS (e.g. Oracle, SQL Server, DB2, Postgres)
  - Document search engines (e.g. Google, Yahoo! Search, Lucene, Ferret)
  - Programmable dataflow engines (e.g. Hadoop MapReduce)
- Ground things in relevant applications

Quiz Question

- Name some widely-used applications

Why take this class?

A. Database systems are at the core of CS
B. They are incredibly important to society
C. The topic is intellectually rich
D. A capstone course for undergrad
~~E. It isn't that much work~~
F. Looks good on your resume

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 managing data
  - complex queries & query semantics*
  - over massive data sets
  - using enormous computing resources
  - maintained over time
- concurrency control for data manipulation
  - controlling concurrent access
  - ensuring transactional semantics
- reliable data storage
  - maintain data semantics even if you pull the plug

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
- 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? This is not a course for:
  - Oracle administrators
  - SQL Server engine developers
    - Though it's useful for both!
- It is a course for well-educated computer scientists
  - Database system concepts and techniques increasingly used "outside the box"
    - Ask your friends at Microsoft, Yahoo!, Google, Apple, etc.
    - Actually, they may or may not realize it!
  - A rich understanding of these issues is a basic and (un?)fortunately unusual skill.

Who?

- Instructor
  - Prof. Joe Hellerstein
  - cs186prof@db.cs.berkeley.edu
- TAs
  - Peter Alvaro
  - Kuang Chen

How? Workload

- Projects with a "real world" focus:
  - Modify the internals of a "real" open-source database system: PostgreSQL
    - Serious C system hacking in a ~500KLoc codebase
    - Measure the benefits of our changes
  - Build web-based applications
    - Using Ruby on Rails, PostgreSQL, Ferret text search
  - Learn to write data-centric coad
    - Using SQL and MapReduce paradigms
- Exams – 2 Midterm & 1 Final

How? Administrivia

- http://inst.eecs.berkeley.edu/~cs186
- Office Hours:
  - JMH:
    - Tues 12:30-1:30
    - Thurs 11:00-12:00 (in 685 Soda)
  - TAs: TBA
- Discussion Sections meet next Monday

How? Administrivia, cont.

- Textbook
  - *Database Management Systems, 3rd Edition*
    - Ramakrishnan and Gehrke
  - *Agile Web Development with Rails, 2nd edition*
    - e-book is fine (better?)
  - *Programming Ruby, 2nd 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
  - Teams of 2
  - Think about this now! Find a partner ASAP.
- Class bulletin board - http://tech.groups.yahoo.com/group/cs186-09/
  - read it regularly and post questions/comments.
- Email:
  - mail to the cs186 course account will not be answered
- Class Blog for announcements

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
  - object-relational mapping using Ruby on Rails
- Today's lecture is from Chapter 1 in R&G
- Read Chapter 2 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, or fields.

Example: University Database

- Schema:

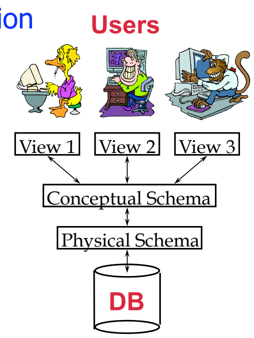
```
- Students(sid text,
           name text,
           login text,
           age integer,
           gpa float)

-  Courses(cid text,
           cname text,
           credits integer)

-  Enrolled(sid text,
            cid text,
            grade text)
```

Levels of Abstraction

- 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 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

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

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 atomic sequence of database actions (reads/writes)
- takes DB from one consistent state to another

consistent state 1 —transaction→ consistent state 2

Example

checking: \$200
savings: \$1000
transaction →
checking: \$300
savings: \$900

- 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 >= 0

Concurrent transactions

- Goal: **execute xacts {T1, T2, … Tn}, 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 <u>some</u> 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)*

Suspose:

1. T1 locks Savings
2. T2 locks Checking

- Now neither transaction can proceed!

– called "deadlock"

– DBMS will <u>abort and restart</u> one of T1 and T2

– Need "undo" mechanism that preserves consistency

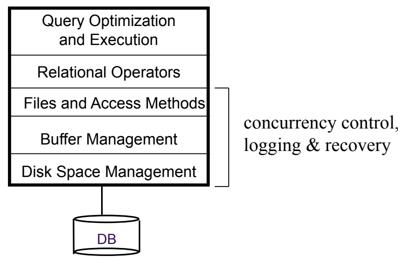
– Undo mechanism also necessary if system crashes between "Savings –= 100" and "Checking += 100" …

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 *<u>log</u>* of all actions carried out by the DBMS:
  - Record all DB modifications in log, *<u>before</u>* they are executed
  - To abort a xact, undo logged actions in reverse order
  - If system crashes, must:
    1) **<u>undo</u>** partially executed xacts (ensures **atomicity**)
    2) **<u>redo</u>** committed xacts (ensures **durability**)
  - *trickier than it sounds!*

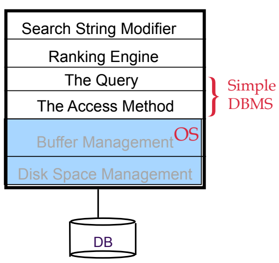
Architecture of a DBMS …

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
  - Can be difficult to write apps above (improving, though)
  - This cost & complexity must be offset by need
  - General-purpose, not suited for special-purpose tasks (e.g. text search, matrix or timeseries data, etc.)

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
- Text search engine
  - similar to relations underneath
  - many "application-specific" smarts

Summary, cont

- Levels of abstraction & data independence.
  - Codd's Data Independence foundation of modern Databases
    - Hellerstein's inequality
  - Classic idea, resonates in the most modern SW

- 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