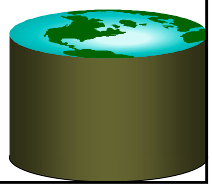
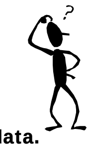
15-415 - Database Applications

Spring Semester 2003
Prof. Anastassia Ailamaki

"Knowledge is of two kinds: we know a subject ourselves, or we know where we can find information upon it."
-- Samuel Johnson (1709-1784)



What Is a Database *System*?



- **Database:**
  a very large, integrated collection of data.
- **Models a real-world *enterprise***
  - Entities (e.g., students, courses)
  - Relationships
    (e.g., Lance Armstrong is *enrolled in* 15-415)
  - More recently, also includes active components
    (e.g. "business logic")
- **A *Database Management System (DBMS)* is a software system designed to store, manage, and facilitate access to databases.**

Is the WWW a DBMS?

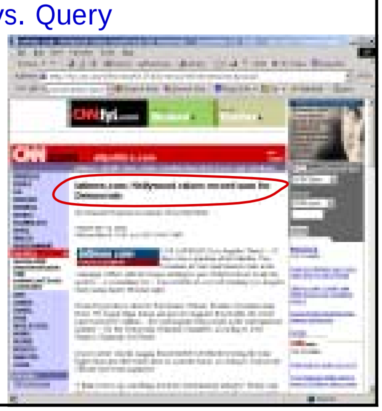
- **Fairly sophisticated search available**
  - crawler *indexes* pages for fast search
- **But, currently**
  - data is mostly unstructured and untyped
  - can't manipulate the data
  - few guarantees provided for freshness of data, consistency across data items, fault tolerance, ...
  - Web sites typically have a DBMS in the background to provide these functions.
- **The picture is quickly changing**
  - New standards like XML can help data modeling
  - Research groups are working on providing some of this functionality *across multiple web sites.*

"Search" vs. Query

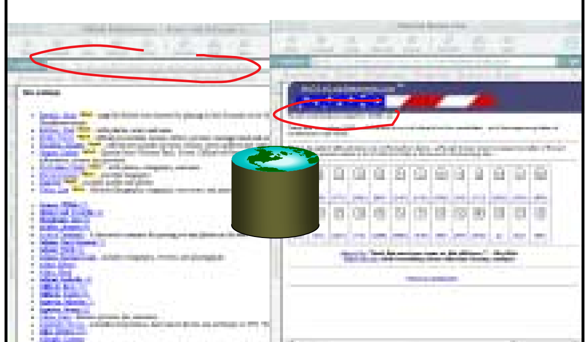
- **What if you wanted to find out which actors donated to Al Gore's presidential campaign?**
- **Try "actors donated to gore" in your favorite search engine.**

"Search" vs. Query

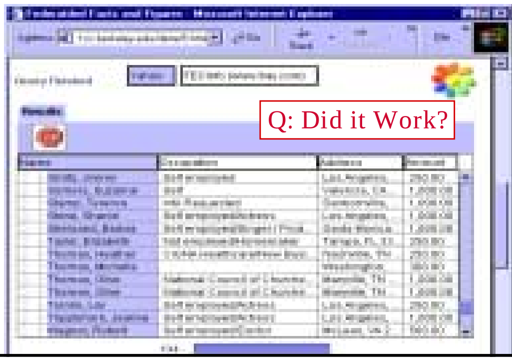
- **"Search" can return only what's been "stored"**
- **E.g., best match at iWon, Google, AskJeeves top ten:**



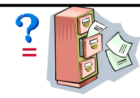
A "Database Query" Approach



“Yahoo Actors” JOIN “FECInfo”
(Courtesy of the Telegraph group)



Is a File System a DBMS?



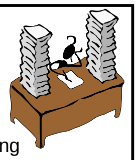
- **Thought Experiment 1:**
  - You and your project partner are editing the same file.
  - You both save it at the same time.
  - Whose changes survive?

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

- **Thought Experiment 2:**
  - You’re updating a file.
  - The power goes out.
  - Which of your changes survive?

A) All B) None C) All Since last save D) ???

Why Study Databases??



- **Shift from *computation* to *information***
  - always true for corporate computing
  - Web made this point for personal computing
  - more and more true for scientific computing
- **Need for DBMS has exploded in the last years**
  - Corporate: retail swipe/clickstreams, “customer relationship mgmt”, “supply chain mgmt”, “data warehouses”, etc.
  - Scientific: digital libraries, Human Genome project, NASA Mission to Planet Earth, sensors
- **DBMS encompasses much of CS in a practical discipline**
  - OS, languages, theory, AI, multimedia, logic
  - Yet traditional focus on real-world apps

What’s the intellectual content?

- **representing information**
  - data modeling
- **languages and systems for querying data**
  - complex queries with real *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

About the course - Workload

- **Projects this semester cover:**
  - DBMS Internals

    (requires systems programming in “C”)
  - Database Query design, optimization and processing
  - Database Applications
- **Other homework assignments and/or quizes**
- **Exams – 1 Midterm & 1 Final**
- **Projects to be done in groups of 3**

YOU WILL NOT RECEIVE A GOOD GRADE IN THE COURSE IF YOUR EXAM SCORES ARE POOR
(Regardless of project grades)

About the Course - Administrivia

- **http://www.cs.cmu.edu/~natassa/15-415**
- **Prof. Office Hours:**
  - Wean Hall 7109, Mondays 3-4pm (except today)
- **TAs: Spiros Papadimitriou, Minglong Shao, Joey Trdinich**
  - Office Hours: (check web page)

About the Course - Administrivia

- **Textbook**
  - Ramakrishnan and Gehrke, 3rd Edition
- **Grading, hand-in policies, etc. on Web Page**
- **Cheating policy: zero tollerance**
- **Class newsgroup: academic.cs.15-415**
  - read it regularly and post questions/comments.
  - mail broadcast to all TAs will not be answered
- **Announcements: academic.cs.15-415.announce**
  - Only Prof. Posts course announcements in this newsgroup

A 15-415 Infomercial

- **A "free tasting" of things to come in this class:**
  - data modeling
  - Query languages
  - file systems & DBMSs
  - concurrent, fault-tolerant data management
  - DBMS architecture
- **Next Time**
  - Database Design using the Entity-Relationship model
- **Today's lecture is from Chapter 1 in R&G**
- **Read Chapter 2 for next class.**

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!

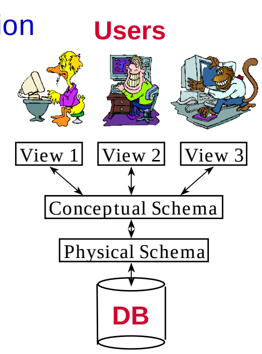
[1] ad hoc: formed or used for specific or immediate problems or needs
[2] SMOP: Small Matter Of Programming

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.

Levels of Abstraction

- **Views describe how users see the data.**
- **Conceptual schema defines logical structure**
- **Physical schema describes the files and indexes used.**
- **(sometimes called the ANSI / SPARC model)**



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.**
- **Q: Why is this particularly important for DBMS?**

Because databases and their associated applications persist.

Concurrency Control

- **Concurrent execution of user programs: key to good DBMS performance.**
  - Disk accesses frequent, pretty slow
  - Keep the CPU working on several programs concurrently.
- **Interleaving actions of different programs: trouble!**
  - e.g., deposit & withdrawal on same account
- **DBMS ensures such problems don't arise: users can pretend they are using a single-user system. (called "Isolation")**
  - Thank goodness!

Transaction: An Execution of a DB Program

- **Key concept is a transaction: an atomic sequence of database actions (reads/writes).**
- **Each transaction, executed completely, must take the DB between consistent states.**
- **Users can specify simple integrity constraints on the data. The DBMS enforces these.**
  - Beyond this, the DBMS does not understand the semantics of the data.
  - Ensuring that a single transaction (run alone) preserves consistency is ultimately the user's responsibility!

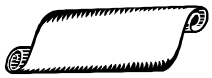
Scheduling Concurrent Transactions

- **DBMS ensures that execution of {T1, ... , Tn} is equivalent to some *serial* execution T1' ... Tn'.**
  - 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 held until the end of the transaction. (Strict 2PL locking protocol.)
  - Idea: If an action of Ti (say, writing X) affects Tj (which perhaps reads X), one of them, say Ti, will obtain the lock on X first and Tj is forced to wait until Ti completes; this effectively orders the transactions.
  - What if Tj already has a lock on Y and Ti later requests a lock on Y? (Deadlock!) Ti or Tj is aborted and restarted!

Ensuring Transaction Properites

- **DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.**
- **DBMS ensures *durability* of *committed* Xacts even if system crashes.**
- **Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of Xacts:**
  - 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. Effects of committed transactions are *redone* using the log.
  - trickier than it sounds!

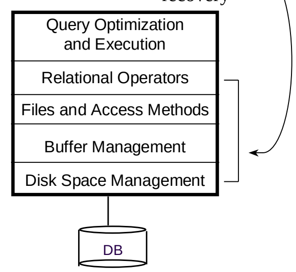
The Log



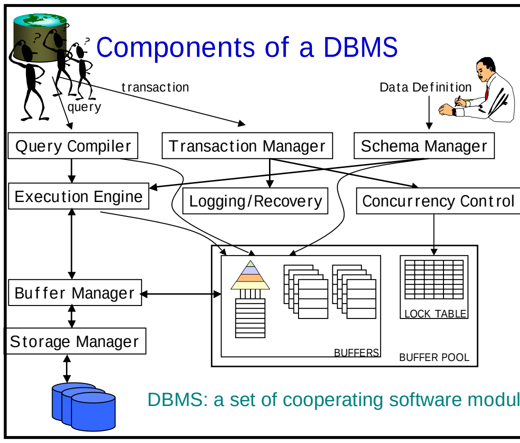
- **The following actions are recorded in the log:**
  - *Ti writes an object*: the old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *Ti commits/aborts*: a log record indicating this action.
- **Log records chained together by Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).**
- **Log is often *duplexed* and *archived* on "stable" storage.**
- **All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.**

Structure of a DBMS

- **A typical DBMS has a layered architecture.**
- **The figure does not show the concurrency control and recovery components.**
- **Each system has its own variations.**
- **The book shows a slightly more detailed version.**



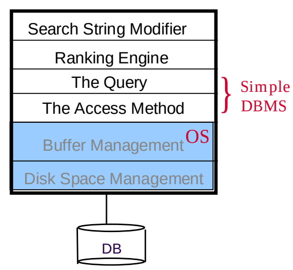
Components of a DBMS



DBMS: a set of cooperating software modules

FYI: 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)
  - no semantics: WYGIWIGY



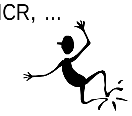
There may be time to talk about some of these text tricks in this class, but it won't be a focus.

Advantages of a 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 ...

- **DBMS vendors, programmers**
  - Oracle, IBM, MS, Sybase, Informix, NCR, ...
- **End users in *many* fields**
  - Business, education, science, ...



- **DB application programmers**
  - Build data entry & analysis tools on top of DBMSs
  - Build web services that run off DBMSs
- **Database administrators (DBAs)**
  - Design logical/physical schemas
  - Handle security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

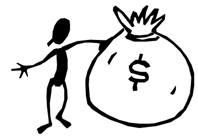
...must understand how a DBMS works

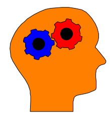
Summary (part 1)

- **DBMS used to maintain, query large datasets.**
  - can manipulate data and exploit *semantics*
- **Other benefits include:**
  - recovery from system crashes,
  - concurrent access,
  - quick application development,
  - data integrity and security.
- **Levels of abstraction provide data independence.**
- **In this course we will explore:**
  1) How to be a sophisticated user of DBMS technology
  2) What goes on inside the DBMS

Summary, cont.

- DBAs, DB developers a key part of the information economy





- DBMS R&D represents a broad, fundamental branch of the science of computation

•NEXT CLASS – Entity Relationship Modeling (read Chapter 2)