CS6093
Advanced Database

Juliana Freire

Instructor Provenance

• Juliana Freire
  – Freire sounds like “Frady” in English
• Research interests
  – Databases, provenance, large-scale information integration, visualization, Web mining, Web crawling, semi-structured data management (XML), management of scientific data, systems
  – http://vgc.poly.edu/~juliana

From Fortaleza to New York City
Undergrad UFC, Brazil (88-91) → MS+PhD @ SUNY Stony Brook, NY (91-97) → Research Scientist @ Bell Labs, NJ (97-02) → Professor @ OGI, Oregon (02-04) → Professor @ Utah (04-11) Founder @ VisTrails Inc (07- ) Professor @ NYU-Poly (11- )
Database Technology: The Present

• Databases are everywhere, even when we don’t see them!
• Why use databases?
  – Avoid inconsistency in the presence of updates and failures; allow concurrent and efficient access to data
• Relational database management systems (RDBMS) are robust, reliable and efficient
  – $15 billion worldwide market!
• 30+ years of databases research – systems and theory
  – Relational model, query optimization, transaction processing – everything you have already learned!

Still in the present, but new trends

• Traditional RDBMS: originally developed for OLTP
  •Data well-designed and follows rigid structure
  •Queries, updates, transactions in a controlled and centralized environment
  •Complex systems - require expertise to set up and maintain
### Still in the present, but new trends

- **Traditional RDBMS:** originally developed for OLTP
- **Explosion in the volume of data:** sensors, satellites, gene sequences, documents, simulations, …
  - Abundant computing power, cheap memory and disk
  - Data available and exchanged over the Web
- **New users:** non-DB experts create and manipulate data
- **New data:** diverse and complex structure, sometimes with no structure
- **New applications:** need operations that go beyond relational operators
- Data is distributed over networks, and owned by autonomous sources
- Data is **shared:** need to keep track of its provenance

### Databases of the Future: Requirements

- **Need to reinvent** the traditional database framework
  - Borrow techniques and ideas from other disciplines, e.g., machine learning, information retrieval
- **Solutions must be flexible**
  - Cater to a wide range of requirements for different applications and data
  - Tools that can be mixed and matched to build custom data management solutions
- **Solutions must be easy to use**
  - Cater to users with different levels of expertise
  - Automate as much as possible – further efforts on autonomic computing
What the course is about?

- Advanced topics in Databases
  - Stuff *not covered* in database textbooks
- Study recent and not so recent database literature related to data management
  - For the Web
  - For scientific data
  - For integrating, analyzing, visualizing data
What the course is not about?

• Transaction processing
• Query optimization
• Data warehousing
• …
• Stuff covered in database textbooks

What the course is about?

• Advanced topics in Databases
  – Stuff not covered in database textbooks
• Study recent and not so recent database literature related to data management
  – For the Web
  – For scientific data
  – For integrating, analyzing, visualizing data
• Practice to review papers
  – Read articles with a critical view
• Identify research problems
• Practice to give presentations
• Do research!

If you are not sure you should take this course, please talk to me!
Readings

• There is no textbook
• This course is based on research papers
  – I will provide links to the papers we will study

Workload

• No exams
• Paper presentations, discussion and position papers
• Final project
  – You can propose a project, or
  – Select from a list I will provide (by next week)
Lectures

- Lectures by instructor and *students*
- Each week we will cover a topic (or a set of sub-topics)
  - I will give an overview of the broad area
  - You will present and we will study different approaches to solving particular problems in the topic
  - I will suggest the material you need to read, and you may also look for additional references

Mechanics

- Two teams will be assigned to each topic (or sub-topic)
- "*Cheerleaders*" will present summary of the topic and present the area in the best possible light
  - Largely derived from the assigned readings—encouraged to go beyond these to discover other interesting work within the same topic
  - The presentation should "not" be a linear presentation of the sections in the papers, instead it should give a general overview of the problem, challenges involved in addressing the problem, existing solutions, and directions for new work in the area.
- "*Discussants*" will present a short rebuttal to the presenters talk
  - Prepare questions, counter-examples, and a generally with a devil's advocate attitude toward the work.
Mechanics

- **“Cheerleaders”** vs. **“Discussants”**
- **Goal**: set up a debate-like atmosphere in which we can argue about the pros and cons of the basic technologies.
- **Scribe volunteer**: will take notes during class, summarize discussion, and post them on the Wiki
- **The rest of the class** expected to actively participate in the debate
  - Everyone who is not a presenter or a discussant will write a brief position paper which captures your own thoughts about the readings.
  - Position papers should be about 1-page long---posted on the Wiki before class

Communication

- Class Web page
  http://vgc.poly.edu/~juliana/courses/cs6093/
- Class Wiki for sharing presentations, papers, notes (TBD)
- Please enter your information at:
  https://docs.google.com/spreadsheet/viewform?formkey=dHhJekdhdf9wZ0J6U0tpX1dtSUc2R3c6MQ
Today

- Brief DB primer
- Introduction to Provenance Management

What Is a DBMS?

- A database is a very large, integrated collection of data.
- Models real-world enterprise.
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS6093)
- A Database Management System (DBMS) is a software package designed to store and manage databases.
Storing Data: Database vs File System

• In the early days, database applications were built on top of file systems
• Drawbacks of using file systems to store data:
  – Data redundancy and inconsistency
    • Multiple file formats, duplication of information in different files
  – Difficulty in accessing data
    • Need to write a new program to carry out each new task
  – Data isolation — multiple files and formats
  – Integrity problems
    • Integrity constraints (e.g., account balance > 0) become part of program code — hard to add new constraints or change existing ones

Storing Data: Database vs File System (cont.)

• Drawbacks of using file systems (cont.)
  – Atomicity of updates
    • Failures may leave database in an inconsistent state with partial updates carried out
    • E.g., transfer of funds from one account to another should either complete or not happen at all
  – Concurrent access by multiple users
    • Concurrent access needed for performance
    • Uncontrolled concurrent accesses can lead to inconsistencies
      – E.g. two people reading a balance and updating it at the same time
  – Security problems
• Database systems offer solutions to all the above problems
• There are still many apps that use the file system…why?
Why use Database Systems?

- Data independence and efficient access.
  - Easy + efficient access through declarative query languages and optimization
- Data integrity and security.
  - Safeguarding data from failures and malicious access
- Uniform data administration.
- Concurrent access, recovery from crashes.
- Reduced application development time.

What's in a DBMS?

<table>
<thead>
<tr>
<th>data model</th>
<th>query language</th>
<th>transactions &amp; crash recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical DB design</td>
<td>SQL</td>
<td>transactions</td>
</tr>
<tr>
<td>relational model</td>
<td>QBE, views</td>
<td></td>
</tr>
<tr>
<td>&quot;above the water&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>what users see</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;below the water&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>how it's built:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mapping to files</td>
<td>query optimiz.</td>
<td>Locking</td>
</tr>
<tr>
<td>clustering, indices</td>
<td>query implem.</td>
<td>concurrency control</td>
</tr>
<tr>
<td>&quot;above the water&quot;</td>
<td></td>
<td>recovery, logs</td>
</tr>
<tr>
<td>&quot;below the water&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Models

- A data model is a collection of concepts for describing data, relationships, semantics and constraints.
- A schema is a description of a particular collection of data, using the 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.

Relational Model

- Example of tabular data in the relational model

<table>
<thead>
<tr>
<th>Customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
<th>account-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
<td>A-101</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td>A-215</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
<td>A-201</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
<td>A-217</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td>A-201</td>
</tr>
</tbody>
</table>
Data Models (cont.)

- Other data models:
  - Entity relationship
  - Object-oriented
  - Semi-structured/XML
  - RDF

Levels of Abstraction

- Many **views**, single **conceptual (logical) schema** and **physical schema**.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.

☞ Schemas are defined using DDL; data is modified/queried using DML.
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.

☞ *One of the most important benefits of using a DBMS!*
Data Definition Language (DDL)

- Specification notation for defining the database schema
  - E.g. `create table account (
    account-number char(10),
    balance integer)
  `  
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - database schema

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language
SQL

• SQL: widely used non-procedural language
  – E.g. find the name of the customer with customer-id 192-83-7465
    
    ```sql
    select customer.customer-name
    from customer
    where customer.customer-id = '192-83-7465'
    ```
  – E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465
    
    ```sql
    select account.balance
    from depositor, account
    where depositor.customer-id = '192-83-7465' and depositor.account-number = account.account-number
    ```

• Application programs generally access databases through one of
  – Language extensions to allow embedded SQL
  – Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database

Examples of Bad Interactions

• You and your spouse each take $100 from different ATM’s at about the same time.
  – The DBMS better make sure one account deduction doesn’t get lost
• You transfer $100 from savings to checking, the system goes down after the $100 is debited from your savings account, but before it is credited to the checking account
  – The DBMS must ensure that the money does not disappear
• Compare: An OS allows two people to edit a document at the same time. If both write, one’s changes get lost.
Transaction: An Execution of a DB Program

- Key concept is transaction, which is an atomic sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
  - Users can specify some simple integrity constraints on the data, and the DBMS will enforce these constraints.
  - Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

Transaction Management

- A transaction is a collection of operations that performs a single logical function in a database application
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.
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.

Storage Management

- Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data
Storage Access

- A database file is partitioned into fixed-length storage units called **blocks**.
  - Blocks are units of both storage allocation and data transfer.
- Database system seeks to **minimize the number of block transfers** between the disk and memory.
  - Reduce the number of disk accesses by keeping as many blocks as possible in main memory.
- **Buffer** – portion of main memory available to store copies of disk blocks.
- **Buffer manager** – subsystem responsible for allocating buffer space in main memory.
  - Different replacement strategies, e.g., LRU, pinned block.
  - LRU: Use past pattern of block references as a predictor of future references.
  - Pinned block: ensure recoverability – block cannot be written while it is being updated.

Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.
Why Use a RDBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Concurrency Control: Issues

- Expensive, and time consuming
  - Any transaction needs to lock the database
- If an application only reads, still pay the price
Why Not Use a RDBMS?

• Require the schema to be define a priori, but for some applications, we don’t know exactly what the data will look like
• Enforces a strict structure, but some data is semi-structured
• Concurrency control is expensive, even when you do not need it!
• It is just too hard to configure…

Databases make these folks happy ...

• End users and DBMS vendors
• DB application programmers
  – E.g., smart webmasters
• Database administrator (DBA)
  – Designs logical/physical schemas
  – Handles security and authorization
  – Data availability, crash recovery
  – Database tuning as needs evolve

Must understand how a DBMS works!
Databases could make these folks happy ...

- People who do not have CS training and who do not understand how a DBMS works
  - Biologists, oceanographers, climate scientists, statisticians
- We are not there yet…

Summary

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- DBAs hold responsible jobs and are well-paid! 😊
- DBMS R&D is one of the broadest, most exciting areas in CS --- many new, important applications that require us to re-think the existing technology.