Database Management Systems

Mirek Riedewald

Some slides based on presentation by Ramakrishnan and Gehrke

Logistics

- Go to http://www.ccs.neu.edu/~mirek/classes/2012-S-CS3200 for all course-related information
- Grading
  - Homework: 50%
  - Midterm: 20%
  - Final exam: 25%
  - Participation: 5%
- TA: Bahar Qarabaqi
- Office hours will be announced soon
- Can always email us with questions or to set up appointments

Project

- Work with a real DBMS: Postgres
- Work with database using SQL and Java (JDBC)
- Deliverables: code and reports
- We have a database server set up
  - You can install client GUI tool to connect to it
- You can also install a server on your own machine
  - We cannot provide support for that

Goals for This Course

- Learn about the fundamentals of relational DBMS
  - Declarative programming: specify WHAT you want, not HOW to get it
  - Set-oriented processing and query optimization
  - Data independence
  - Transactions and recovery from crashes
- Be able to create, access, and manipulate a database through SQL and from an application
- Work with a real DBMS
  - Acquire enough background to more quickly become an expert on any other DBMS
- Be better able to understand and critically evaluate features of competing data management offerings

What This Course Cannot Do

- Make you a DB admin or SQL guru
  - Requires a lot of practice, but you will get the basics
- Make you an expert on the DBMS from vendor XYZ
  - Course provides general fundamentals, future employers can train you for their specific environment
- Provide details about DBMS internals
  - That's a whole different course

Any Questions So Far?

What Is a DBMS?

- Database = very large, e.g., terabytes, integrated collection of data.
  - Entities (e.g., students, courses)
  - Relationships (e.g., Joe is taking CS 3200)
- Database Management System (DBMS) = software package designed to store and manage databases.

Why Study Databases?

- Ubiquitous in enterprises and daily life
  - ATMs, banking, retail transactions, flight booking, customer databases
- Shift from computation to information
  - Simplify data management tasks
  - Enable efficient data processing at large scale
- Datasets increasing in diversity and volume.
  - Digital libraries, Human Genome project, Sloan Digital Sky Survey
- DBMS encompasses most of CS
  - OS, languages, theory, AI, multimedia, logic

Files vs. DBMS

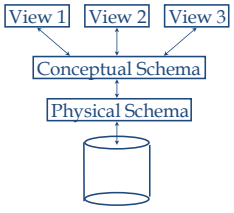
- File example: Find all young customers (age < 25) in a large customer file
  - Solution 1: simple sequential scan of entire file
  - Solution 2: if the file is already sorted by age, scan from beginning and stop when first "old" customer is found
    - Note: sorting a file costs more than scanning it completely
  - Solution 3: if an index on age exists, it directly points to the right customers in the file
    - This is only efficient if the number of young customers is a small fraction
  - Best solution depends on data properties (fraction of young customers), query properties (age range selected), and physical data layout (sorted file, index)
  - Once your program finally works, what if data layout or file size changes…?
- Writing code for managing very large files is difficult
  - Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access)
- Protect data from inconsistency due to multiple concurrent users
- Crash recovery, security, access control,…

Data Models

- Data model = collection of concepts for describing data.
- Schema = description of a particular collection of data, using a given data model.
- The relational data model 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

- 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.



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

- One of the most important benefits of using a DBMS
- Applications insulated from how data is structured and stored
- Logical data independence: Protection from changes in logical structure of data
  - If logical structure changes, create view with old structure
  - Works fine for queries, but might be tricky for updates
- Physical data independence: Protection from changes in physical structure of data
  - Query and update logical structure, not physical structure

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent and relatively slow, the CPU can work on several user programs concurrently
- Interleaving actions of different user programs can lead to inconsistency
  - E.g., check is cleared while account balance is being computed
- DBMS ensures such problems do not arise: users and programmers can pretend they are using a single-user system

Transaction = Atomic DB Program

- Transaction = 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 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!

Ensuring Atomicity

- DBMS ensures atomicity (all-or-nothing property) even if system crashes in the middle of a Xact
- 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
  - After a crash, the effects of partially executed transactions are undone using the log

Databases And Startups

- DBMS perfect as data management system for startups
- LAMP stack: Linux OS, Apache Web server, MySQL DBMS, PHP (or Perl, Python)
- Why LAMP?
  - The price is right
  - Easy to code using MySQL and scripting language
  - Easy to deploy
    - Set up LAMP on laptop, build app locally, then deploy on the Web
  - Ubiquitous hosting
    - Even cheapest Web hosting options usually allow running PHP, MySQL

Example: eBay

- 1995—1997: GDBM (GNU library of DB functions)
- 1997—1999: Oracle (biggest DBMS vendor)
- 1999—2001: still Oracle, but now multiple servers
- 2001—present: split DBs by functionality, pull most functionality from DBMS up into application layer
- DBMS still important component
  - Initially the data management entity, scaling well…
  - …until eBay grew so much that customized solutions were needed
  - DBMS is general-purpose, and extreme challenges require more customized solutions

NoSQL Movement

- Growing popularity of non-relational data stores
  - Document stores, key-value stores, eventually consistent stores, graph DB, object-oriented DB, XML DB
- Examples: MongoDB, CouchDB, Google's BigTable, Amazon's Dynamo
- Many of them driven by performance challenges
  - Inherent tradeoff between consistency, availability, and tolerance to network partitions (Eric Brewer, UC Berkeley)
    - Maintaining consistent state across 100s of machines requires expensive agreement (communication)
    - Failures reduce availability, unless consistency is weakened (1000 machines => failures happen all the time)
- Solutions: weaker consistency guarantees or tailored solution for specific workload

MapReduce vs. DBMS

- Google's answer to data processing challenges
- Programming paradigm for distributed computation on large clusters
- Two phases
  - Map: map each input record independently to a set of (key, value) pairs
  - Reduce: process set of all values with the same key together
- Read what two DBMS luminaries think and how readers reacted
  - http://databasecolumn.vertica.com/database-innovation/mapreduce-a-major-step-backwards/
  - http://databasecolumn.vertica.com/database-innovation/mapreduce-ii/
  - Links seem broken now, but a snapshot of their content will be on Blackboard
- Active research area in databases to combine best of both worlds

Exciting Times

- Worldwide relational DBMS software revenue $15.2B in 2006 (source: Gartner)
  - Dominant players: Oracle, IBM, Microsoft, Teradata
- Smaller companies with specialized data management solutions
  - Vertica, Greenplum, Netezza, and many more
- Virtually every enterprise relies on DBMS
- Close relative of data warehousing
- Mushrooming of noSQL alternatives and parallel/distributed data management solutions
- Knowing the principles of relational DBMS is essential for understanding these trends.

Summary

- DBMS used to maintain and query large datasets
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security
- Levels of abstraction give data independence
- DBMS R&D is a broad and very exciting area in CS