Databases Uses and Introduction to the Relational Model

20 October 2009
Lecture 1

Course Introduction

- Topics: Databases, Database Management Systems, and SQL
- Prerequisites:
  - 1-02-113: Introduction to Information Systems Engineering (may be taken at the same time)
  - 1-02-220: File Organization and Processing
  - 1-02-284: Logic
- Instructor: Michael May
  - Ph.D. University of Pennsylvania, Philadelphia

Course Administrivia

- Lecture
  - Meets
  - Number of Lectures: 13
  - Lecture schedule is in the Syllabus on Telem
  - Course Web Page:
- Targil
  - Meets
  - Number of Review/Lab sessions: 13
- Text Books
  - Ramakrishnana, Raghu and Johannes Gehrke. ***Database Management Systems***. McGraw Hill. 3rd Edition
  - Silberschatz, Abraham, Henry F. Korth, and S. Sudarshan. ***Database System Concepts***. McGraw Hill. 5th Edition
  - Ullman, Jeffrey D. and Jennifer D. Widom. ***First Course in Database Systems***. Prentice Hall. 2nd Edition

In Class

- Attendance is recommended
  - Attendance will be taken from time to time
  - Not attending will not directly affect your grade
  - It will likely affect your grade in other ways
- Decorum is required
  - Talking or disturbing class will not be tolerated
  - I turn off my cell phone for class – **YOU SHOULD TOO**
  - If you must talk – **LEAVE CLASS**

Grading

- Components of the course
  - Quizzes – 4%
    - 4 of them. Lowest one dropped.
  - Assignments – 18%
  - Project – 18%
  - Final Exam – 60%

Targilim

- Targil will consist of:
  - Exercising what we learned
  - Asking/Answering questions
  - Trying out queries on MS SQL Server
  - Working on Project
- No grade for anything done during Targil

Assignments

- Four assignments over the course of the semester
  - May work in groups of up to 3 students
- Will exercise concepts and skills explained in class:
  - ERD
  - Schema development
  - Basic SQL
  - More advanced SQL – views, stored procedures
- Will help you in working on your projects

Semester Project

- Design and implement a database and interactive GUI front end using the tools and techniques learned in the classroom
  - Can be done in groups of 2
- Developed in phases over the course of the semester
  - Whole thing graded at the very end
  - Meet with me after each phase submitted
- Final project submission
  - Project Report
  - Project Presentation on TBA in Targil

Exams

- Final examination in February
  - Worth 60% of the grade
  - Everything we learned
  - Everything you should have read

Let's begin…

Topics for Today

- Managing Data
- Why use a database?
- Describing and Storing Data
- Queries
- Transaction Management
- Structure of a DBMS

- Sources:
  - Ramakrishnan and Gehrke 1.1-1.9

Managing Data

- Databases are designed to enable easy management of large quantities of related information
- We will talk about in this course:
  - Database Design
  - Database Analysis
  - Using Databases
  - Concurrency and Robustness
  - Efficiency and Scalability

Some History

- The first DBMS was designed by GE in 1960s
- IBM Developed Information Management Systems (IMS) in 1960s
  - Still In use – SABRE airline reservation system
- Edgar Codd proposed *Relational Data Model*
  - Foundation for most DBMS today
  - We'll talk about changes to that model next semester
- IBM developed System R in the 1980s – turned into SQL

Today

- SQL:
  - Standardized by ANSI and ISO
  - Relatively uniform, but some features are not implemented by various vendors
- Later:
  - Data Warehouses: 1980s and 1990s – rich data and storage specialization
  - Enterprise Resource Planning (ERP)
  - Management Resource Planning (MDP)
  - Backend for web programs – PHP, Java, Apache, Javascript

Motivating a DBMS

- For *large* amounts of structured data – say 500GB
  - Won't fit in memory!
  - To retrieve it you need something just to deal with the data files
  - Manage moving it on and off disk
  - Concurrency?
- Those are the major parts of a DBMS

DBMS Advantages

- Data Independence
  - Don't worry how it's stored, retrieved, updated
- Efficient Data Access
  - Manage large quantities, distributed
- Data Integrity and Security
- Data Administration
  - Centralized and uniform management

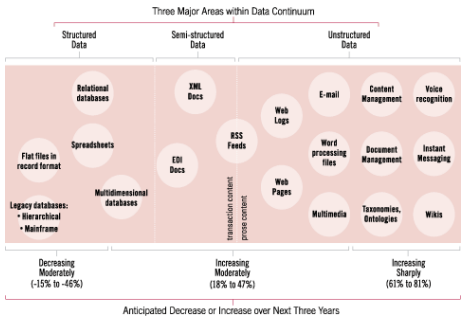
DBMS Advantages

- Concurrent Access
- Crash Recovery
  - When media fails, power fails
- Reduced Application Development Time
  - Don't worry about managing files, reads
- It is the backend for your applications

Don't use a Database when:

- Data is not structured
- Need to do complex manipulations
- Small quantity of data
- Need to manage file or byte level

Examples of Unstructured Data



So Far

- Managing Data
- Why use a database?
- Describing and Storing Data
- Queries
- Transaction Management
- Structure of a DBMS

Describing and Storing Data

- Data is stored based on the *Data Model*
  - We'll talk about the *Relational Data Model*
  - There are others: hierarchical, OO
- We can abstract up to a higher level – the *Semantic Model*
  - Describes how data relates to other data
  - We'll talk about the *Entity-Relationship* semantic model

What is the Relational Model?

- Relations: A set of records
  - Usually unordered
  - Some way to identify each record (a key)
- Records: A structured n-tuple of fields (columns)
- *Schema* for a relation is the structure of its records

Example Schema

- Students (*sid:*string, *name*:string, *login*:string, *age*:integer, *gpa:*real)

Sid	*Name*	*Login*	*Age*	*Gpa*
53	Jones	ajones@cs	18	34.5
54	Jones	bjones@cs	30	91.3
12	Smith	smith@is	23	78.2
53	Cohen	cohen@math	19	80.2

What makes this useful?

- It's just a table?
- We can do more:
  - Identify rows by some set of fields (key)
  - Constraints on data ranges, types, integrity
  - Link one relation to another
  - Enforce that changing one automatically changes another

Other options?

- We didn't need to use tables
- Object based databases (next semester)
- Hierarchical databases
  - Old, but getting back in vogue with XML
  - That's another course

Levels of Abstraction

- Relational model introduces levels of abstraction
  - Lets us ignore details of lower levels
  - Each level has its own schema, written in terms of the level below
- Three important levels:
  - Physical
  - Conceptual
  - External

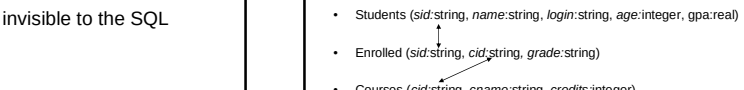
Write their schemas with the *Data Definition Language* (SQL)

Physical Schema

- How data is organized on disk
  - Relations usually stored as unordered collections of records
  - Indexes are used to make sorting and searching faster
    - We'll talk a bit about them next semester
- The physical schema is normally invisible to the SQL coder
  - We won't get to it in this course

Conceptual Schema

- Describes data as *entities* and *relationships* or links between relations
  - A student is enrolled in a course



External Schema

- How the database is seen from the outside
  - Written in terms of the conceptual schema
- What interface or relations are shown to the outside world?
  - May just be the original relations
  - Or may be composite relations built from other relations in the schema – this is called a *View*

Views and External Schema

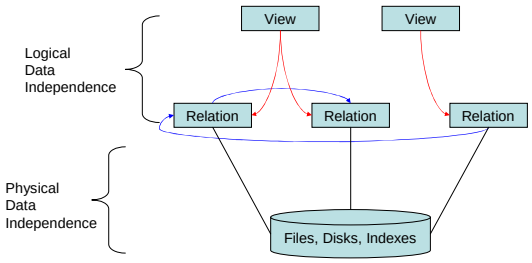
- A (Dynamic) View is a virtual relation calculated solely based on the contents of other relations. Each time it is accessed, it is recalculated before use.
- Instead of Students – Enrolled – Courses, we can define summaries of the data
- Enrollment (*cid*:string, *enrollment*:integer)
  - Shows only how many students per course
- GpaPerCourse(*cid*:string, *avggpa*:float)
  - Shows the average grade for students in the course

Why Views?

- Data abstraction
  - The logistics people don't need student names, just the count
- Data hiding
  - The lecturer shouldn't need to know each student's grade average, just the average for the class
- Data protection
  - Logistics people can't update the enrollment figures, just view them
    - How else could we enforce this?
- Data updates
  - Just update the base relations

Data Independence

- One nice outcome of the levels of abstraction – data independence



Summary

- Managing Data
- Why use a database?
- Describing and Storing Data