CSE 3241: Database Systems I
Databases Introduction
(Ch. 1-2)

Jeremy Morris
Outline

- What is a database?
- The database approach
  - Advantages
  - Disadvantages
- Database users
- Database concepts and System architecture
What is a database?

- A collection of related data
  - ..but not just a random collection of data
    - The “My Documents” folder on your hard drive isn’t a database, even though the data in it might all be “related”
    - Though you could turn your “My Documents” folder into a database with a bit of work
What is a database?

- A collection of related data
  - ..but not just a random collection of data
- A collection of *logically coherent* data with some *inherent meaning* designed and built for some *specific purpose*
  - Logically coherent
    - Pieces of data are connected to each other in a rational, logical manner
  - Inherent meaning
    - The data as a whole can be described (e.g. “a collection of customers”)
  - Specific purpose
    - The data was organized with a particular audience in mind
What is a database?

- Simple example: A telephone book
  - A collection of names and associated telephone numbers

- More complex example: Library card catalog
  - A collection of information on the various books that the library owns

- Another complex example: Netflix catalog
  - A collection of information on the movie selection offered by Netflix, including customer reviews
What a database is not

- A database is a logical collection of data
  - Not that nowhere in this definition are “computers” or “software” mentioned
  - Databases are ancient
    - Clay tablets recording customer transactions
    - “Old-school” library card catalogs (on actual cards)
    - Grandma’s recipe box
    - … all of these are databases
Computerized databases have come to dominate the idea of databases

- Computers allow:
  - Better organization
  - Better ability to search
  - Ability to minimize redundancies in your data
  - Ability to put your database on the web
  - ... and many other benefits

- When we talk about database software, we are talking about *Database Management Systems* (DBMS)
  - DMBS theory and practice will be the focus of this course
  - More specifically *Relational DBMS* (RDBMS)
Database Management System (DBMS)

- From the book: “a general purpose software system that enables the defining, constructing, manipulating and sharing of databases”
  - **Defining**
    - Specifying what should be in the database
    - **Meta-data**: description of what the data in the database means
  - **Constructing**
    - Storing data in the database
    - Handling file creation and organization on the disk
  - **Manipulating**
    - Querying and updating the database
      - **Query**: retrieving specific data from the database
      - **Update**: change the contents of the database
  - **Sharing**
    - Allowing users to access the database
Simplified view of a database system
Consider a simple database of information for a university

Design phases:
1. Requirements Specification
2. Conceptual Design
3. Logical Design
4. Physical Design
Sample Requirement: Data

- What information does it need to hold?
  - Student information
    - Names, Ids, Majors, Year
    - What courses they’ve taken, what grades they’ve gotten
  - Course information
    - Names, Course Numbers
    - Any prerequisites required?
    - Individual sections (Instructor, when it’s offered, room #, etc.)
Sample Requirement: Queries

- What kinds of questions will we expect this database to be able to answer?
  - What is John Smith’s grade?
  - What courses is John Smith taking?
  - Who are all of the students who took the CSE 670 course under Professor Morris in Spring 2011?
    - What grade did they get?
  - What are the times of all of the sections of CSE 670 in Spring of 2011?
    - Who teaches them?
Sample Requirement: Maintenance

What kinds of changes should we expect to be making regularly to the data?

- John Smith is now a Senior instead of a Junior
- John Smith received an ‘A’ in CSE 670 in Spring 2011
- Enroll John Smith in Prof. Morris’s CSE 670 class for Spring 2011
- Add a section of CSE 670 for Autumn 2011
- Cancel a section of CSE 670 in Spring 2011
Sample Requirement: Constraints

- What kinds of checks do we need to put into place?
  - Does each student have a unique student ID?
    - “Key constraint”
  - Does the section refer to a course that actually exists?
    - “Referential constraint”
  - Does the student have the pre-requisites for the class he is trying to enroll in?
    - “Semantic constraint”
Conceptual Design

- Break the information down into different collections of data:
  - Student information
  - Course information
  - Section information
  - Grade information

- Determine the relationships between the various collections
  - Each section is of a specific course
  - Each grade is tied to a student and a course
Sample Logical Design

- Determine the different data elements to be stored in each record
  - Including their *data types*
  - Each student will have:
    - Name (String)
    - Id (Number)
    - Major (String)
    - Year (Number)
  - Each course will have a name, number, hours, etc.

- Logical design determines how the records in the database will be organized
Sample logical design for a UNIVERSITY database (Elmasri & Navathe, pg. 7-9)
The Database Approach

- Traditional file processing
  - Each application developer comes up with his own file format for each specific application
    - There may be some sharing of formats, but data is generally locked to an application

- Database approach
  - A single repository of data
    - Defined once
    - Accessed by multiple applications
    - Accessed by multiple users
Database Approach - Characteristics

- Self-describing
- Separation of programs and data
- Multiple views of the data
- Sharing of data
  - Multiple users
  - Multiple applications
Self-describing

- Every database system contains a description of the database structure
  - Meta-data (also known as a database catalog)
    - “Data about data”
    - Description of what the data in the database means
Separation of programs and data

- Traditional file approach
  - Data definition encoded in the application programs

- Database approach
  - Use meta-data to advertise what’s available to applications and users
  - Known as data abstraction
    - Don’t build your applications to read from a fixed file format
    - Build your applications to use a conceptual representation of the data provided by a data model
      - DBMS provides software to act as intermediary between the data model and the physical data stored on disk
Multiple views of the data

- Different users require different subsets of the data to be visible
  - A customer at Amazon.com needs to see his shopping cart
  - A warehouse worker for Amazon.com needs to see a list of all the orders that need to be fulfilled from his warehouse

- Restricted subsets of data in the database is known as a view
  - Allows a level of access restriction
Sharing of data

- The database approach allows multiple users to share data simultaneously
  - No worries about locked files, or only one user accessing the system at a time
- Transaction processing
  - Multiple users updating the database do so in a “sane” way
  - A set of accesses against a database is a transaction
  - Transactions are ACID – Atomic, Consistent, Isolated and Durable
    - Atomic – transactions are “all or nothing”
    - Consistent – transactions must not put the database in a state that violates its constraints
    - Isolated – transactions occur as if they happened by themselves
    - Durable – completed transactions can survive system failure
Advantages of the DBMS Approach

- Controlling redundancy
  - Design databases to store each logical item only once

- Controlling access

- Efficient query and update processing
  - Everything in one place, rather than scattered around

- Backups!

- Enforcing integrity constraints
  - Data types
  - Referential integrity
    - Integrity across records in different tables
Advantages of the DBMS Approach (cont)

- Enforcing standards
- Reduced application development time
  - Most of your development work is already done once you’ve built the database
- Flexibility
  - Can change the database without changing every single application
- Instant updates to information
Disadvantages

- Overhead!
  - High initial monetary investment in hardware, software
  - Security, referential integrity, concurrency control and other features don’t come for free
    - Higher hardware costs than flat files

- Consider these factors before going with a DBMS approach
  - Cost
  - Is this going to be a single-user database?
  - Do you really need all the features of a DBMS?
  - Is speed more important (embedded devices)?
Database Users

- **End users**
  - Typical (or “naïve”) users
    - Most users of a database
    - Use canned screens and reports to do their job
  - Sophisticated (or “power”) users
    - Need better understanding of the database
    - Write their own reports – often for one-off jobs
Database Users

- Systems Analysts & Software Engineers
  - Determine what software the users need
  - Write it for them

- Database Administrators (DBA)
  - Administer access rights, monitor the database for performance, etc.

- Database Designers
  - Design the layout of the database
    - Often overlap with DBAs or Systems Analysts depending on the organization
Database Concepts (Terminology!)

- Databases provide *data abstraction*
  - Keep the data separate from how it is stored in the system
    - Data actually stored in files, as with any other computer system
    - Database approach provides a way of letting the user interact with the data without caring about how it is stored

- We use a *data model* to provide this abstraction
  - Model provides the database structure
    - Data types
    - Relationships between data records
    - Constraints on data records
Data Models

- High-level model (conceptual model)
  - How do users perceive the data?
    - Entities
      - real-world concepts or objects (Student)
    - Attributes
      - some property of an entity (GPA)
    - Relationship – association between two entities
      - “Students” are enrolled in “sections”

- Low-level model (physical data model)
  - How is the data actually stored in the computer?
Data Models (continued)

- Representational model (implementation model)
  - Midway between high-level and low-level
    - Still a way to organize the data so that it can be understood by users
    - But also still related to how the data is physically stored
  - Relational data model
    - A widely-used representational data model
    - Will be the focus of this course
- Other models
  - Network model, hierarchical model
    - Legacy models – not used as much these days
  - Object data model
Database schema

- Part of the data model
- Description of database
  - Use a *schema diagram* to display database schema

```
STUDENT
Name | Student_number | Class | Major

COURSE
Course_name | Course_number | Credit_hours | Department

PREREQUISITE
Course_number | Prerequisite_number

SECTION
Section_identifier | Course_number | Semester | Year | Instructor

GRADE_REPORT
Student_number | Section_identifier | Grade
```
Three schema architecture

- Schemas at three different levels:
  - Internal level
    - Physical storage of the database
  - Conceptual level
    - Hides details of physical storage
    - Describes entities, relationships, constraints, etc.
    - Usually the level where the database is actually designed and implemented
  - External level
    - User views
    - Describes parts of database a user is interested in
    - Hides the rest
      - Different users, different views

- Most DBMSs do not completely separate these levels
  - Support it to one extent or another

- Data only at internal level
  - Each level provides mappings to levels above and below it
Data Independence

- **Logical data independence**
  - Change the conceptual schema without changing external schemas or application programs
    - Expand the database
    - Change constraints
    - Reduce the database

- **Physical data independence**
  - Change the physical storage of the data without changing the conceptual schema
    - Physical file reorganization (directory restructuring)
Database Languages

- Variety of languages historically used in databases:
  - Data Definition Language (DDL)
    - Used to define “Conceptual Schema” level
  - Storage Definition Language (SDL)
    - Used to define “Internal Schema” level
  - View Definition Language (VDL)
    - Used to define “External Schema” level
  - Data Manipulation Language (DML)
    - Used to perform retrievals, updates, deletes

- Most modern DBMS’s do not treat these as separate languages
  - Use a single comprehensive language (such as SQL)
    - Covers all of these areas of functionality