

Database Management Systems: Introduction



Spring 2009

Instructor: Bhagi Narahari
narahari@gwu.edu

Office Hours: M,W: 3-5:00pm, Tues 5-6pm
TA Lab: Peter Letzler

Introduction to DBMS

What Is a Database?



- ❖ A very large, integrated collection of data.
 - Not arbitrary, unrelated data
- ❖ Models real-world enterprise.
 - Entities (e.g., students, courses)
 - Relationships (e.g., Jimmy Page is taking CS178)

Why Study Databases??



- ❖ Nothing better to do on Mon, Wed 12-45-2pm!

Why Databases?

Most CS courses concentrate on code – our interest is managing and representing **data**

Warning: this course doesn't focus on teaching SQL or how to be an Oracle DBA (though it will get you started)

... So what in the world are we studying for 14 weeks???

Why Databases??



- ❖ Information gathering is first step to analysis
 - > **tons of data can be collected easily using current technology**
- ❖ To effectively analyze data, must
 - > **collect relevant data**
 - > **store in manner amenable to efficient access**
 - > **provide infrastructure for ease of programming**
- ❖ Data analysis methods are current emphasis in the market

Data Analysis

- ❖ Data Warehousing
 - > **nothing but a big database (remember: you can charge your client more if you say warehouse instead of DB!!!).**
- ❖ Web search engines
 - > **Get data from the web**
 - > **Create index and provide fast search methods**
 - > **Size of data can be huge**

Data Analysis: Data Mining

- ❖ Data mining: finding 'hidden patterns' in data; i.e., patterns and relationships that are not 'obvious'
 - > **purchasing patterns of supermarket customers**
 - **Pattern: 40% of Customers who buy beer also buy diapers.**
 - > **How do you use the above pattern/knowledge to improve your marketing strategy ??**
 - Leave it to the Business Majors to worry about!!
- ❖ Data mining is "engine" behind Personalization software

Databases...?

- ❖ Why the discussion on Data mining etc?
 - Analysis is important to make informed decisions
 - efficient analysis requires efficient storage&design
 - efficient storage&design requires study of DBMS!
- ❖ Data Mining and other data analysis tools
 - require background in database design and analysis!!
- ❖ and remember- DBMS is basic backbone in Transaction Processing systems!
 - Airline reservations, banking, e-commerce

Some salient points about Databases

- ❖ Most CS courses concentrate on code – our interest in this course: managing,, manipulating and representing data.
 - This course does not focus on making you into an expert on Oracle, etc.
- ❖ DBMS encompasses most of CS
 - OS, languages, theory, “A”I, multimedia, logic
 - Brings together various concepts from different CS topics
 - Operating systems, prog languages, networks, security, data structures, algorithms, etc.

Database & Application Development Process

- ❖ Organize and Store relevant data
- ❖ Operate on the data
 - Search, aggregate,...
- ❖ Present results to user
 - Provide interface
- ❖ Analyze data
 - Extract patterns ...

Example: PDA

- ❖ Information on your PDA/Cell-phone
- | Calendar | Event | Day | When | Who | Where |
|----------|-------|------|------|------|-------|
| | lunch | 1/17 | 1pm | John | TGIF |
-
- | Contacts | Who | Phone | Email | Address |
|----------|------|----------|---------|------------|
| | John | 123-4567 | john@.. | 1 main st. |

Example:

- ❖ What if we want to include contact info on our calendar?
 - Do we also have to keep tel. numbers, email, etc.
 - Should we expand the number of "fields"?
 - Should we re-enter the data?

Solution

- ❖ Can we link calendar with contacts?
 - "link" calendar entries with contact information and show the results of the two
 - Link should be based on something - simple solution is to link on person's name
 - What is name is not unique?
 - How to follow links?

Data Models and data representation

- ❖ All of the data we've seen have an implicit data model
 - Basic assumption on what is an item of data, how to interpret it, etc.
- ❖ The relational model was the first model of data that is **independent** of its data structures and implementation
 - A theory of normalization guides you in designing relations

Data Models

- ❖ A *data model* is a collection of concepts for describing data.
 - Starting point to design of DBMS
- ❖ 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.
- ❖ Other data models:
 - Network, Hierarchical, OO

Organizing information

- ❖ Person has attributes
 - > Ssn
 - > GW ID
 - > Name
 - > ..
- ❖ Student IS a person who:
 - > Takes courses at GW
 - > Is given grades
 - > registers.
 - > ..
- ❖ This is yet another kind of information.....
 - > Where have you seen this ?

How to organize the information ?

- ❖ What is the data needed ?
 - > Eg: What do we need to store to uniquely identify a student entity ?
- ❖ How to store & organize the data ?
 - > How many attributes are really needed
- ❖ This is why we need to study schema design
 - > Normal forms

What Is a DBMS?



- ❖ Database is a large, cohesive, collection of data.
 - > Not arbitrary, unrelated data
 - > Models real-world enterprise.
- ❖ A Database Management System (DBMS) is the software to store/retrieve and manage databases.
 - > Provides an interface over the database

The DBMS Provides an Interface over the Database

- ❖ A database management system (DBMS) is a software package designed to store and manage databases
 - > **Reliable storage & recovery** of 100s of GB
 - > **Querying/ updating** interface and API (for applications and Web pages)
 - > Support for many **concurrent** users
- ❖ Why do we need a DBMS, instead of coding in Java?

DBMS Benefit #1: Generality and Declarativity

- ❖ Don't require the programmer or user to know details like indices, sort orders, machine speeds, disk speeds, concurrent users, etc.
- ❖ Instead, the programmer/user programs with a *logical model* in mind
- ❖ The DBMS "makes it happen" based on an understanding of relative costs of different methods

Benefit #2: Efficiency and Scale

- ❖ Size of personal address book is probably less than 100 entries, but there are things we'd like to do quickly and efficiently:
 - > "Give me all appointments on 10/28"
 - > "When am I next meeting Jim?"
- ❖ "Program" these as quickly as possible (and make them resilient to data format changes)
- ❖ Scale to a corporate calendar with hundreds of thousands of entries

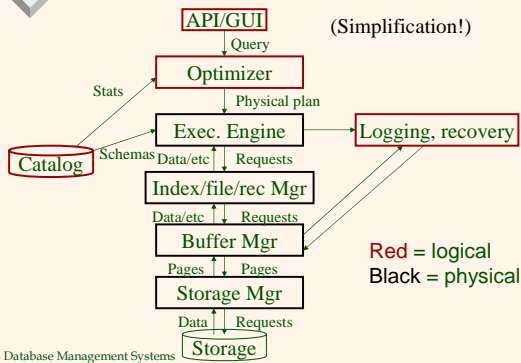
Benefit #3: Management of Concurrency and Reliability

- ❖ Suppose other people are allowed access to your calendar and are allowed to modify it? How do we stop two people changing the file at the same time and leaving it in a physical (or logical) mess?
- ❖ Suppose the system crashes while we are changing the calendar. How do we recover our work?
- ❖ This requires a basic concept...

Transactions

- ❖ Key concept for concurrency is that of a **transaction**: an atomic sequence of database actions (read/write) on data items (e.g. calendar entry).
- ❖ Key concept for recoverability is that of a **log**: keeping track of all actions carried out by the db.

The Layers of the DBMS



The Database Abstraction Provided by the DBMS

We think of databases at two levels:

- > **Logical** structure:
 - What users/programmers see - program or query interface
- > **Physical** structure:
 - Organization on disk, indices, etc.

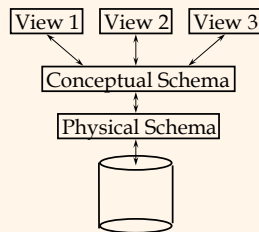
The logical level is further split into:

- > Overall database design (conceptual; seen by the DB designer)
- > **Views** that various users get to see

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 Data Definition Language (DDL);
- * data is modified/queried using Data Manipulation Lang(DML).

Data Independence

A user of a relational database system should be able to use the database without knowing about how the precisely how data is stored,

e.g.

```
SELECT When, Where
FROM Calendar
WHERE Who = "Jane"
```

After all, you don't worry IEEE floating-point when you do division in a Java program or with a calculator

More on Data Independence

Logical data independence

Protects the user from changes in the logical structure of the data:

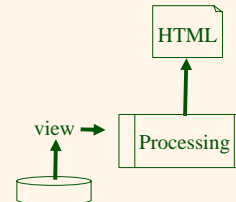
could reorganize the calendar “schema” without changing how we query it

Physical data independence

Protects the user from changes in the physical structure of data:

could add an index on who (or sort by when) without changing how the user would write the query, but the query would execute faster (query optimization)

Presentation Layer (4th Tier): Data-Driven Web Sites



- ❖ “Data driven web sites” also add an HTML “presentation” layer on top of what we’ve seen
- ❖ Or they use XML plus “style sheets” to get the same effect

An Issue: 80% of the World’s Data is Not in a DB!

Examples:

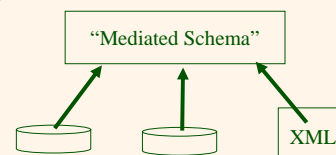
- scientific data (large images, complex programs that analyze the data)
- personal data
- WWW and email (some of it is stored in something resembling a DBMS)

Data management is expanding to tackle these problems

- ❖ Flexibility - data management imposes many constraints to make problems solvable
- ❖ Must deal with entities outside our control

In this course, we’ll start by focusing on databases, but eventually look “outside the box” at the Web and at **gluing together data from many places**

Combining Databases with Mediators (a kind of middleware)



A layer above the three-tiered architecture, to combine multiple databases/sources on the Web

- Some of these are databases over which we have no control
- Some must be accessed in special ways
- We generally need to think about how to translate between different database formats

How to define and use the database: Data Definition and Manipulation Languages

- ❖ data definition language (DDL) to specify database schema
- ❖ Data manipulation language (DML) allows users to access or manipulate data as organized by data model
 - procedural DMLs: require user to specify what data and how to get it
 - non-procedural DMLs: require user to specify what data is needed without specifying how to get it.
 - Commercial languages - SQL

Query Languages

- ❖ Formal query languages: **Relational algebra**, Relational Calculus, Domain calculus
- ❖ Commercial query languages: **SQL**, QUEL
- ❖ SQL: “descendent” of SEQUEL; mostly relational algebra and some aspects of relational calculus
 - has procedural and non-procedural aspects

DBMSs in the Real World

- A huge industry for 20% of the world’s data!
- ❖ Big, mature relational databases
 - IBM, Oracle, Microsoft
 - ❖ “Middleware” above these
 - SAP, PeopleSoft, dozens of special-purpose apps
 - ❖ “Application servers”
 - ❖ Integration and warehousing systems
 - ❖ Current trends:
 - Web services; XML everywhere
 - Smarter, self-tuning systems

Summary

- ❖ DBMS used to maintain, query large datasets.
- ❖ Levels of abstraction give data independence.
 - A DBMS typically has a layered architecture.
- ❖ DBAs hold responsible jobs and are **well-paid!** 😊
- ❖ DBMS form essential piece in information processing applications
 - Data mining, search engines, Human Genome



Course Administration...

Course Outline: Topics

- ❖ Introduction to Relational DBMS
- ❖ Part 1: Logical level design of Relational Databases
 - Formal Query Languages: Rel. algebra
 - Query languages: SQL
 - Relational Schema Design and Normal Forms, Tuning
 - *Start working on Project*
- ❖ Part 2: Physical Database Design
 - Storage, Indexing, File Structures
 - Query Processing and optimization-i.e., how things work
 - Concurrency and Recovery; Intro to transaction processing
 - Overview of Performance modelling
- ❖ Advanced Topics : Security and Privacy, GIS, Information retrieval systems, Data Mining/analysis

Objectives

- ❖ Concepts of data storage and retrieval
- ❖ Working with relational database systems
- ❖ Design and deploy a large database application

- ❖ Outcomes:
 - Fluency in SQL and database application dev.
 - Software integration experience and team experience

Course Information - URL

- ❖ All course material will be placed at the URL:
 - www.seas.gwu.edu/~bhagiweb/cs178/
 - Also linked from my homepage
- ❖ All course announcements will be placed on web-- check once a week!
- ❖ Textbook:
 - Ramakrishnan and Gehrke. *Database Systems*, 3rd Edition.
 - You can use any other Database textbook

Course Requirements: Grading

- ❖ Homeworks + Programming Assignments 35%
 - Homeworks are not programming assignments
 - Programming assignments
 - Some in-class exercises during lab sessions
 - Using MySQL, Oracle, and multi-tier apps using PHP and JDBC
- ❖ late submissions invoke penalty of 10% each day late; and zero points if you submit after solutions are posted
- ❖ Two Exams (in class, closed book): 40%
- ❖ Projects (demos required): 25%

Project

- ❖ A large project requirement with a set of “applications” will be posted on the web site
 - Objective: Build a complete end-to-end working project
- ❖ Three Phase project
 - Phase 1: each student assigned to an application, and required to provide a design of the database schema.
 - Phase 2: One set of teams build application A, Second set of teams build application B
 - Phase 3: Create new teams, with members from group A and B. **Integrate** the applications to provide a complete solution.
- ❖ A clear set of “minimum” requirements will be specified - note that meeting minimum requirements only implies a C grade on the project.
- ❖ Clear deadlines for specific phases and steps in the project will be posted.

Lab Sections and TA

- ❖ Lab sections conducted by TAs
- ❖ Lab sections will cover
 - Intro MySQL, Oracle: SQL, PL/SQL, Forms, JDBC
 - Short tutorials - including application development using PHP, JDBC, XML
 - Clarifications on Programming Assignments
 - Help with analysis of Project (but not in the design of the project)
 - In-class exercises in some weeks: have to implement the queries during the lab; no extensions!
- ❖ You **MUST** request MySQL account by emailing root@seas.gwu.edu - mention that it is for CS178.
- ❖ Detailed Lab schedule coming soon

Academic Integrity Policy

- ❖ www.cs.gwu.edu/academics/integrity.html
 - details and FAQ
- ❖ No collaboration (of any sort) on homeworks
 - Including external resources
- ❖ No collaboration among teams
 - within team each team member must have clear role -- i.e., clearly partitioned tasks for each team member
- ❖ violation of integrity policy -- default is maximum punishment (at least F for course)

Prerequisites

- ❖ CS 143 Programming and Data Structures
- ❖ Languages: Java is required (can do project using PHP, MySQL)
- ❖ CS 135/156 Architecture and Operating system basics
- ❖ CS 123 Discrete Math/Logic

Next . .

- ❖ Read Chapter 1,2
- ❖ Fill out survey
- ❖ Request MySQL account