VirginiaTech

CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #1: Introduction

Many slides based on material by Profs. Murali, Ramakrishnan and Faloutsos

Course Information

- **Instructor**
  B. Aditya Prakash, McBryde 122C, badityap@cs.vt.edu
  – Office Hours: 10-11:30am Mondays and Wednesdays
  – Include string **CS 4604** in subject

- **Teaching Assistant**
  Qianzhou Du, McBryde 106, qiand12@vt.edu
  – Office Hours: 10am-12noon Tuesdays and Thursdays

- **Class Meeting Time**
  Mondays, Wednesdays, Fridays, 9:05-9:55am, Lavery Hall 330

- **Keeping in Touch**
  Course web site http://courses.cs.vt.edu/~cs4604
  updated regularly through the semester
  – *Piazza link on the website*

Textbook

- **Required**

  A First Course in Database Systems, Ullman and Widom, Prentice Hall. (3rd Ed).

  Web page for the book (with errata)

  http://www-db.stanford.edu/~ullman/fcdb.html

- *Optional*:
  - Ramakrishnan and Gehrke, 3rd Ed.
  - Silberschatz, Korth and Sudarshan, 6th Ed.

Pre-reqs and Force-adds

- Prerequisites: a grade of C or better in CS 3114, senior standing
  - every student must fill out **a pre-requisite form**, and must return it **to me at the end of the class** in order to remain enrolled
- Force-add requests:
  - Please fill out the **add form as well**, and return **to me at the end of the class**
  - We (=me or the dept) will let you know hopefully by Friday

Course Grading

Homework	30%	5–6
Midterm exam	15%	(Tentative) March 8, Wed., in class
Final exam	25%	May 11, Saturday, 1:05pm-3:05pm
Course project	30%	6-7 assignments

- Project is spread over 6-7 deliverables
- Projects and homework assignments alternate
- Submit hard copies of homeworks and project assignments at the start of class on the due date (see late policy on website)
- Each class has required reading (on course web page)
- No Pop-Quizzes ☺

Course Project

- Project overview

  http://courses.cs.vt.edu/~cs4604/Spring13/project/project.html
- 2, or 3 persons per project.
- Project runs the entire semester with regular assignments and a final implementation assignment.

Why Study Databases?

- ***Academic***
  - Databases involve many aspects of computer science
  - Fertile area of research
  - Three Turing awards in databases
- ***Programmer***
  - a plethora of applications involve using and accessing databases
- ***Businessman***
  - Everybody needs databases => lots of money to be made
- ***Student***
  - Get those last three credits and I don't have to come back to Blacksburg ever again!
  - Google, Oracle, Microsoft, Facebook etc. will hire me!
  - Databases sound cool!
  - ???

What Will You Learn in CS 4604?

- Implementation
  - How do you build a system such as ORACLE or MySQL?
- Design
  - How do you model your data and structure your information in a database?
- Programming
  - How do you use the capabilities of a DBMS?

- CS 4604 achieves a balance between
  - a firm theoretical foundation to designing moderate-sized databases
  - creating, querying, and implementing realistic databases and connecting them to applications

Course Goals and Outcomes

- Take an English language description and convert it into a working database application.
- Create E/R models from application descriptions.
- Convert E/R models into relational designs.
- Identify redundancies in designs and remove them using normalization techniques.
- Create databases in an RDBMS and enforce data integrity constraints using SQL.
- Write sophisticated database queries using SQL.
- Understand tradeoffs between different ways of phrasing the same query.
- Implement a web interface to a database.

Course Outline

- Weeks 1–5, 13: Query/ Manipulation Languages
  - Relational Algebra
  - Data definition
  - Programming with SQL
- Weeks 6–8: Data Modeling
  - Entity-Relationship (E/R) approach
  - Specifying Constraints
  - Good E/R design
- Weeks 9–13: Relational Design
  - The relational model
  - Converting ER to "R"
  - Normalization to avoid redundancy
- Week 14–15: Students' choice
  - Practice Problems
  - XML
  - Query optimization
  - Data mining

What is the goal of a DBMS?

- Electronic record-keeping

  Fast and ***convenient*** access to information

- DBMS == database management system
  - `Relational' in this class
  - data + set of instructions to access/manipulate data

What is a DBMS?

- Features of a DBMS
  - Support massive amounts of data
  - Persistent storage
  - Efficient and convenient access
  - Secure, concurrent, and atomic access

- Examples?
  - Search engines, banking systems, airline reservations, corporate records, payrolls, sales inventories.
  - New applications: Wikis, social/biological/multimedia/scientific/geographic data, heterogeneous data.

Features of a DBMS

- Support **massive** amounts of data
  - Giga/tera/petabytes
  - Far too big for main memory
- **Persistent** storage
  - Programs update, query, manipulate data.
  - Data continues to live long after program finishes.
- **Efficient** and **convenient** access
  - Efficient: do not search entire database to answer a query.
  - Convenient: allow users to query the data as easily as possible.
- **Secure**, **concurrent**, and **atomic** access
  - Allow multiple users to access database simultaneously.
  - Allow a user access to only to authorized data.
  - Provide some guarantee of reliability against system failures.

Example Scenario

- Students, taking classes, obtaining grades
  - Find my GPA
  - <and other ad-hoc queries>

Obvious solution 1: Folders

- Advantages?
  - Cheap; Easy-to-use

- Disadvantages?
  - No ad-hoc queries
  - No sharing
  - Large Physical foot-print

Obvious Solution++

- Flat files and C (C++, Java…) programs
  – E.g. one (or more) UNIX/DOS files, with student records and their courses

Obvious Solution++

- Layout for student records?
  - CSV ('comma-separated-values')

```
Hermione Grainger,123,Potions,A
Draco Malfoy,111,Potions,B
Harry Potter,234,Potions,A
Ron Weasley,345,Potions,C
```

Obvious Solution++

- Layout for student records?
  - Other possibilities like

```
Hermione Grainger,123        123,Potions,A
Draco Malfoy,111             111,Potions,B
Harry Potter,234             234,Potions,A
Ron Weasley,345              345,Potions,C
```

Problems?

- inconvenient access to data (need ‘C++’ expertize, plus knowledge of file-layout)
  - data isolation
- data redundancy (and inconsistencies)
- integrity problems
- atomicity problems
- concurrent-access problems
- security problems
- …….

Problems-Why?

- Two main reasons:
  - file-layout description is buried within the C programs and
  - there is no support for transactions (concurrency and recovery)

DBMSs handle exactly these two problems

Example Scenario

- RDBMS = "Relational" DBMS
- The relational model uses relations or tables to structure data
- ClassList relation:

Student	Course	Grade
Hermione Grainger	Potions	A
Draco Malfoy	Potions	B
Harry Potter	Potions	A
Ron Weasley	Potions	C

- Relation separates the logical view (externals) from the physical view (internals)
- Simple query languages (SQL) for accessing/modifying data
  - Find all students whose grades are better than B.
  - SELECT Student FROM ClassList WHERE Grade >"B"

A Brief History of DBMS

- The earliest databases (1960s) evolved from file systems
  - File systems
    - Allow storage of large amounts of data over a long period of time
    - File systems do not support:
      - Efficient access of data items whose location in a particular file is not known
      - Logical structure of data is limited to creation of directory structures
      - Concurrent access: Multiple users modifying a single file generate non-uniform results
    - Navigational and hierarchical
    - User programmed the queries by walking from node to node in the DBMS.

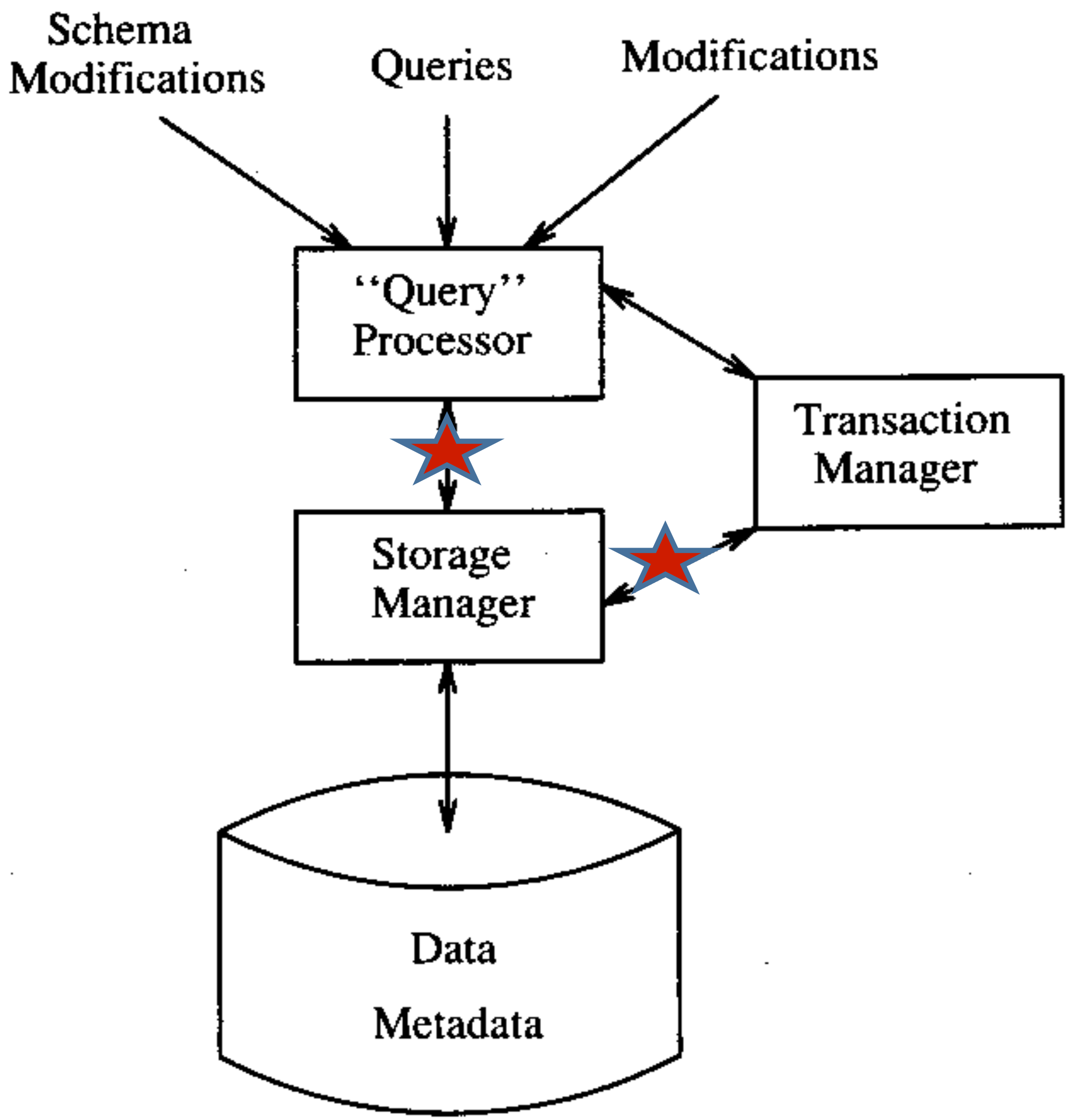
- Relational DBMS (1970s to now)
  - View database in terms of relations or tables
  - High-level query and definition languages such as SQL
  - Allow user to specify what (s)he wants, not how to get what (s)he wants

- Object-oriented DBMS (1980s)
  - Inspired by object-oriented languages
  - Object-relational DBMS

The DBMS Industry

- A DBMS is a software system.
- Major DBMS vendors: Oracle, Microsoft, IBM, Sybase
- Free/Open-source DBMS: MySQL, PostgreSQL, Firebird.
  - Used by companies such as Google, Yahoo, Lycos, BASF….
- All are "relational" (or "object-relational") DBMS.
- A **multi-billion dollar** industry

DBMS Architecture



Transaction Processing

- One or more database operations are grouped into a "transaction"
- Transactions should meet the "ACID test"
  - **A**tomicity: All-or-nothing execution of transactions.
  - **C**onsistency: Databases have consistency rules (e.g. what data is valid). A transaction should NOT violate the database's consistency. If it does, it needs to be *rolled back.*
  - **I**solation: Each transaction must appear to be executed as if no other transaction is executing at the same time.
  - **D**urability: Any change a transaction makes to the database should persist and not be lost.