CSE 544 Principles of Database Management Systems

Lecture 1 - Introduction and the Relational Model

Outline

- Introduction
- Class overview
- Why database management systems (DBMS)?
- The relational model

Course Staff

- Instructor: Dan Suciu
  - Office hours: Wednesday 3:30pm-4:20pm (or by appointment)
  - Location: CSE 662
- TA: Qingda Wen
  - 5th Year Master's student
  - Office hours and location: Fridays 1:30-2:20, CSE 218

About Me

- PhD from UPenn
- Bell Labs / AT&T Labs
- @UW (since 2000)

- I like to combine theory with database systems:
  - Probabilistic databases, causality in data
  - Novel/optimal query processing
  - Data pricing

Goals of the Class/Class Content

- Relational Data Model
  - Data models, data independence, declarative query language.
- Relational Database Systems
  - Storage, query execution and optimization, transactions
  - Parallel data processing, column-oriented db etc.
- Transactions
  - Optimistic/pessimistic concurrency control
  - ARIES recovery system
- Provenance

A Note for Non-Majors

- For the Data Science option: take 414
- For the Advanced Data Science option: take 544

- 544 is an *advanced* class, intended as an introduction to data management research
- Does not cover fundamentals systematically, yet there is an exam testing those fundamentals

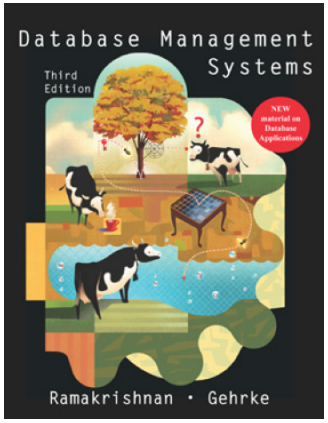
- Unsure? Look at the short quiz on the website.

Class Format

- Two lectures per week: Monday, Wednesday 1:30-2:50
- Mostly lecture, some discussions

Readings and Notes

- Background readings from the following book
  - Database Management Systems. **Third Ed**. Ramakrishnan and Gehrke. McGraw-Hill. [recommended]



- Readings are based on papers
  - Mix of old seminal papers and new papers
  - Papers will be available on class website
- Lecture notes (the slides)
  - Posted on class website after each lecture

Class Resources

- Website: lectures, assignments

  http://www.cs.washington.edu/544

  Project and paper review info to be added

- Mailing list on course website

- Discussion board: discuss assignments, papers, etc

Evaluation

- Assignments 30%
- Exam 30%
- Project 30%
- Paper reviews + class participation 10%

Assignments – 30%

- **HW1**: Use a DBMS
- **HW2**: Datalog
- **HW2**: Build a simple DBMS
- **HW3**: Data analysis in the cloud

- See course calendar for deadlines
- We will accept late assignments with ***very*** valid excuse

Exam – 30%

- March 12, 2:30-4:20

Project – 30%

- Topic
  - Choose from a list of mini-research topics
  - Or come up with your own
  - Can be related to your ongoing research
  - Can be related to a project in another course
  - Must be related to databases / data management
  - Must involve either research or significant engineering
  - Open ended

- Final deliverables
  - Short conference-style paper (6 pages)
  - Conference-style presentation or posters depending on groups

Project – 30%

- Dates will be posted on course website
  - **M1**: form groups
  - **M2**: Project proposal
  - **M3**: Milestone report
  - **M4**: Poster presentation
  - **M5**: Project paper

- More details will be on the website, including ideas & examples

- We will provide feedback throughout the quarter

Paper reviews – 10%

- Between 1/2 page and 1 page in length
  - Summary of the main points of the paper
  - Critical discussion of the paper
  - Guidelines on course website

- Reading questions
  - For some papers, we will post reading questions
  - Address these questions in your reviews

- Grading: credit/no-credit
  - Must submit review 12 HOURS BEFORE lecture
  - Individual assignments (but feel free to discuss paper with others)

Class Participation

- Because
  - We want you to read & think about papers throughout quarter
  - Important to learn to discuss papers
- Expectations
  - Ask questions, raise issues, think critically
  - Learn to express your opinion
  - Respect other people's opinions
- Most students get full credit for class participation, but I may penalize students who miss lectures or just don't participate

Now onward to the world of databases!

Let's get started

- What is a database?
  - A collection of files storing related data
- Give examples of databases
  - Accounts database; payroll database; UW's students database; Amazon's products database; airline reservation database
  - Your ORCA card transactions, Facebook friends graph, past tweets, etc

Data Management

- **Entities**: employees, positions (ceo, manager, cashier), stores, products, sells, customers.
- **Relationships**: employee positions, staff of each store, inventory of each store.
- What operations do we want to perform on this data?
- What functionality do we need to manage this data?

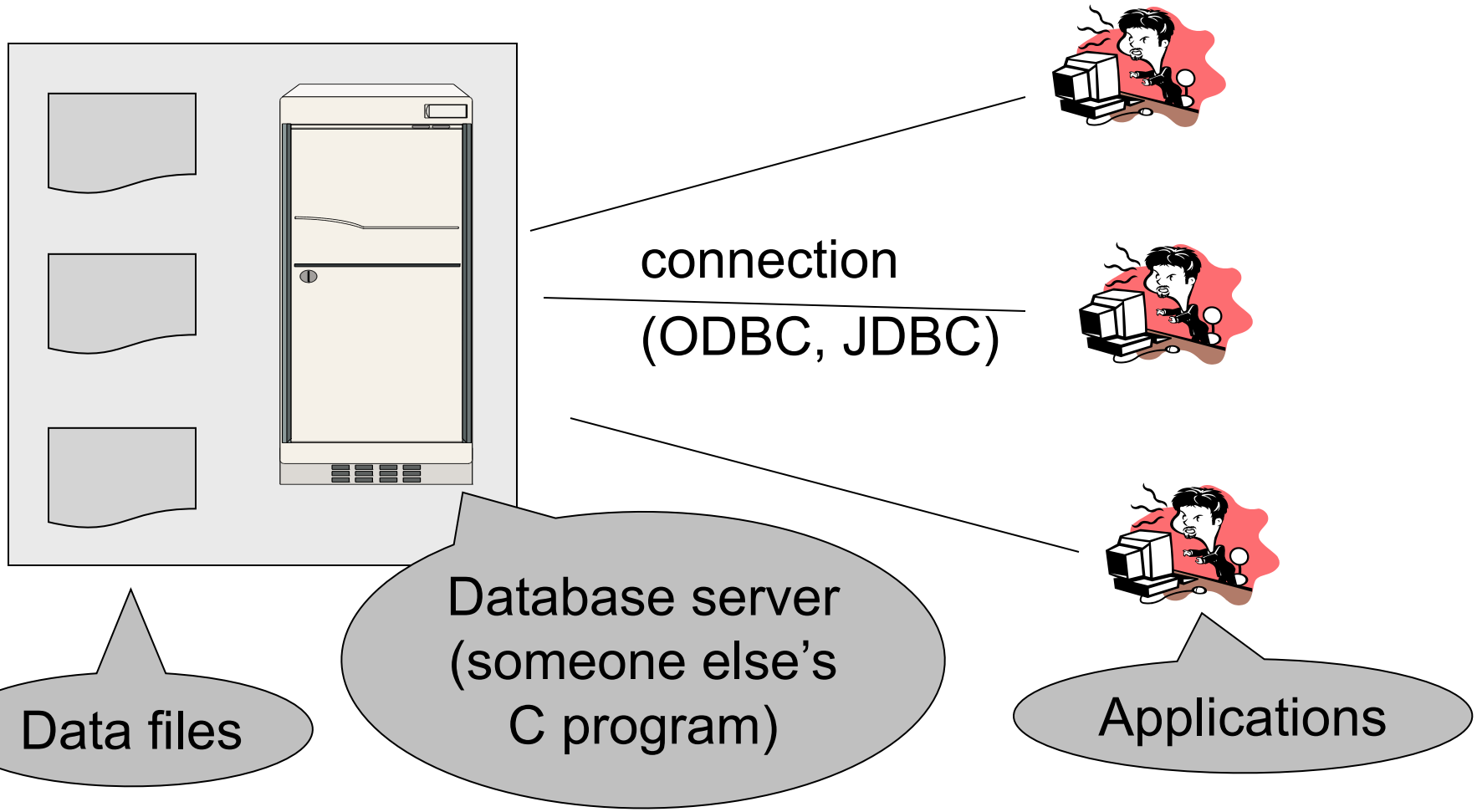
Database Management System

- A DBMS is a software system designed to provide data management services

- Examples of DBMS
  - Oracle, DB2 (IBM), SQL Server (Microsoft),
  - PostgreSQL, MySQL,…

Typical System Architecture

"Two tier system" or "client-server"



Why should *you* care?

- Most of CS and Science today is data driven
- Your research will involve some data component – need to know how to use a DBMS
- Your research may involve some innovative data management solution – need to be up to date with what is known, beyond a DBMS

Main DBMS Features

- Data independence
  - Data model
  - Data definition language
  - Data manipulation language
- Efficient data access
- Data integrity and security
- Data administration
- Concurrency control
- Crash recovery

When not to use a DBMS?

- Main reason: because you didn't take a good DB class!
- Other reasons:
  - DBMS is optimized for a certain workload
  - Some applications may need different data model, or different operations, or a few time-critical operations
  - Example: highly optimized scientific simulations

Outline

- Introductions
- Class overview
- Why database management systems (DBMS)?
- The relational model

Data Model

An abstract mathematical concepts that defines the data

Data models:

- Relational (this course)
- Semistructured (XML, JSon, Protobuf)
- Graph data model
- Object-Relational data model

Relation Definition

- **Database is collection of relations**

- Relation is a table with rows & columns
  - SQL uses the term “table” to refer to a relation

- Relation R is subset of $S_1 \times S_2 \times \ldots \times S_n$
  - Where $\mathbf{S_i}$ is the domain of attribute **i**
  - **n** is number of attributes of the relation

Example

- Relation schema

  Supplier(sno: integer, sname: string, scity: string, sstate: string)

- Relation instance

sno	sname	scity	sstate
1	s1	city 1	WA
2	s2	city 1	WA
3	s3	city 2	MA
4	s4	city 2	MA

sno is called a key
(what does it mean?)

Discussion of the Relational Model

- Relations are *flat* = called 1st Normal Form

- A relation may have a key, but no other FD's = either 3rd Normal form, or Boyce Codd Normal Form (BCNF) depending on some subtle details

[discuss on the white board]

Other Models: Semistructured

- E.g. you will encounter this in HW1:

```
<article mdate="2011-01-11" key="journals/acta/GoodmanS83">
    <author>Nathan Goodman</author>
    <author>Oded Shmueli</author>
    <title>NP-complete Problems Simplified on Tree Schemas.</title>
    <pages>171-178</pages>
    <year>1983</year>
    <volume>20</volume>
    <journal>Acta Inf.</journal>
    <url>db/journals/acta/acta20.html#GoodmanS83</url>
    <ee>http://dx.doi.org/10.1007/BF00289414</ee>
</article>
```

Integrity Constraints

- Condition specified on a database schema
- Restricts data that can be stored in db instance
- DBMS enforces integrity constraints
- E.g. domain constraint, key, foreign key

Constraints are part of the data model

Key Constraints

- **Key constraint:** "certain minimal subset of fields is a unique identifier for a tuple"

- **Candidate key**
  - Minimal set of fields
  - That uniquely identify each tuple in a relation

- **Primary key**
  - One candidate key can be selected as primary key

Foreign Key Constraints

- Field that refers to tuples in another relation
- Typically, this field refers to the primary key of other relation
- Can pick another field as well (but check documentation)

Key Constraint SQL Examples

```
CREATE TABLE Part (
   pno integer,
   pname varchar(20),
   psize integer,
   pcolor varchar(20),
   PRIMARY KEY (pno)
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer,
   PRIMARY KEY (sno,pno)
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer,
   PRIMARY KEY (sno,pno),
   FOREIGN KEY (sno) REFERENCES Supplier,
   FOREIGN KEY (pno) REFERENCES Part
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer,
   PRIMARY KEY (sno,pno),
   FOREIGN KEY (sno) REFERENCES Supplier
                          ON DELETE NO ACTION,
   FOREIGN KEY (pno) REFERENCES Part
                          ON DELETE CASCADE
);
```

General Constraints

- Table constraints serve to express complex constraints over a single table

```
CREATE TABLE Part (
  pno integer,
  pname varchar(20),
  psize integer,
  pcolor varchar(20),
  PRIMARY KEY (pno),
  CHECK ( psize > 0 )
);
```

- It is also possible to create constraints over many tables