Introduction to Database Systems
CSE 444

Lecture 1

Introduction

Staff

- Instructor: Hal Perkins
  - CSE 548, perkins@cs.washington.edu
    Office hours: CSE labs tba, office drop-ins and appointments welcome
- TA: Michael Ratanapintha
  - michaelr at cs
  - Office hours: tba

Communications

- Web page: http://www.cs.washington.edu/444
  - Lectures, homework, projects will be available there
- Discussion list
  - See the web page
  - Discussions about the course, databases, etc. Stay in touch outside class
- Mailing list
  - Mostly announcements, intent is fairly low traffic
  - You are already subscribed if you are registered

Textbook

Main textbook, available at the bookstore:

- *Database Systems: The Complete Book, 2nd ed.*, Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom

You will get the most out of class if you read (skim / get confused about) related sections before seeing them in lecture

Other Texts

Available at the Engineering Library

(not on reserve – would anyone care if they were?):

- *Database Management Systems*, Ramakrishnan
- *Fundamentals of Database Systems*, Elmasri, Navathe
- *Foundations of Databases*, Abiteboul, Hull, Vianu
- *Data on the Web,* Abiteboul, Buneman, Suciu

Course Format

- Lectures MWF, 10:50-11:50 am, EE 037
- Quiz sections: Th 9:40-10:40 or 10:50-11:50, EE 045

- 4 Mini-projects
- 3 homework assignments

- Midterm and final

Grading

- Homeworks 30%
- Mini-projects 30%
- Midterm 20%
- Final 20%*

*During summer, the final exam is the last day of class. Roughly a 2nd midterm.

Four Mini-Projects

1. SQL (already posted)
2. SQL in Java
3. Database tuning
4. Parallel processing: MapReduce

Due: Wednesdays every other week, online, 11pm

Three Homework Assignments

1. Conceptual Design
2. Transactions
3. Query execution and optimization

Due: Wednesdays every other week, also 11 pm

Late Policy

- You have 4 late days to use during the quarter however you wish
  - No more than 2 on any single assignment or project
  - Used in 24 hour chunks
  - No other late assignments accepted

    (And we may specify no late days for particular assignments if needed to hand out solutions before exams or at the end of the quarter)

Academic Conduct

- We all learn best when we work with others, talk to colleagues, etc., and you definitely should do that, **but…**
- Anything you submit for credit is expected to be your individual work (or your group's work if the assignment specifically allows for that)
  - Enough said?

Outline of Today’s Lecture

1. Overview of a DBMS

2. A DBMS through an example

3. Course content

Database

What is a database ?

Give examples of databases

Database Management System

What is a DBMS ?

Give examples of DBMSs

Required Data Management Functionality

1. Describe real-world entities in terms of stored data
2. Create & persistently store large datasets
3. Efficiently query & update
    1. Must handle complex questions about data
    2. Must handle sophisticated updates
    3. Performance matters
4. Change structure (e.g., add attributes)
5. Concurrency control: enable simultaneous updates
6. Crash recovery
7. Security and integrity

DBMS Benefits

- Expensive to implement all these features inside the application
- DBMS provides these features (and more)
- DBMS simplifies application development

How do we decide what features should go into the DBMS?

Market Shares

From 2007 Gartner report:

- IBM: 21% market with $3.2BN in sales
- Oracle: 47% market with $7.1BN in sales
- Microsoft: 17% market with $2.6BN in sales

An Example

The Internet Movie Database

http://www.imdb.com

- Entities:
  Actors (1.8M), Movies (1.5M), Directors, …
- Relationships:
  who played where, who directed what, …

Tables

Actor:

id	**fName**	**lName**	**gender**
195428	Tom	Hanks	M
645947	Amy	Hanks	F
. . .			

Cast:

pid	**mid**
195428	337166
. . .	

Movie:

id	**Name**	**year**
337166	Toy Story	1995
. . .	. . .	. ..

SQL

```
SELECT *
FROM  Actor
```

SQL

```
SELECT count(*)
FROM  Actor
```

This is an *aggregate query*

SQL

```
SELECT *
FROM  Actor
WHERE lname = ‘Hanks’
```

This is a *selection query*

SQL

```
SELECT *
FROM  Actor, Casts, Movie
WHERE lname='Hanks' and Actor.id = Casts.pid
  and Casts.mid=Movie.id and Movie.year=1995
```

This query has *selections* and *joins*

1.8M actors, 11.4M casts, 1.5M movies – how can it be so fast?

How Can We Evaluate the Query ?

Actor:

id	fName	lName	gender
. . .		Hanks	
. . .			

Cast:

pid	mid
. . .	
. . .	

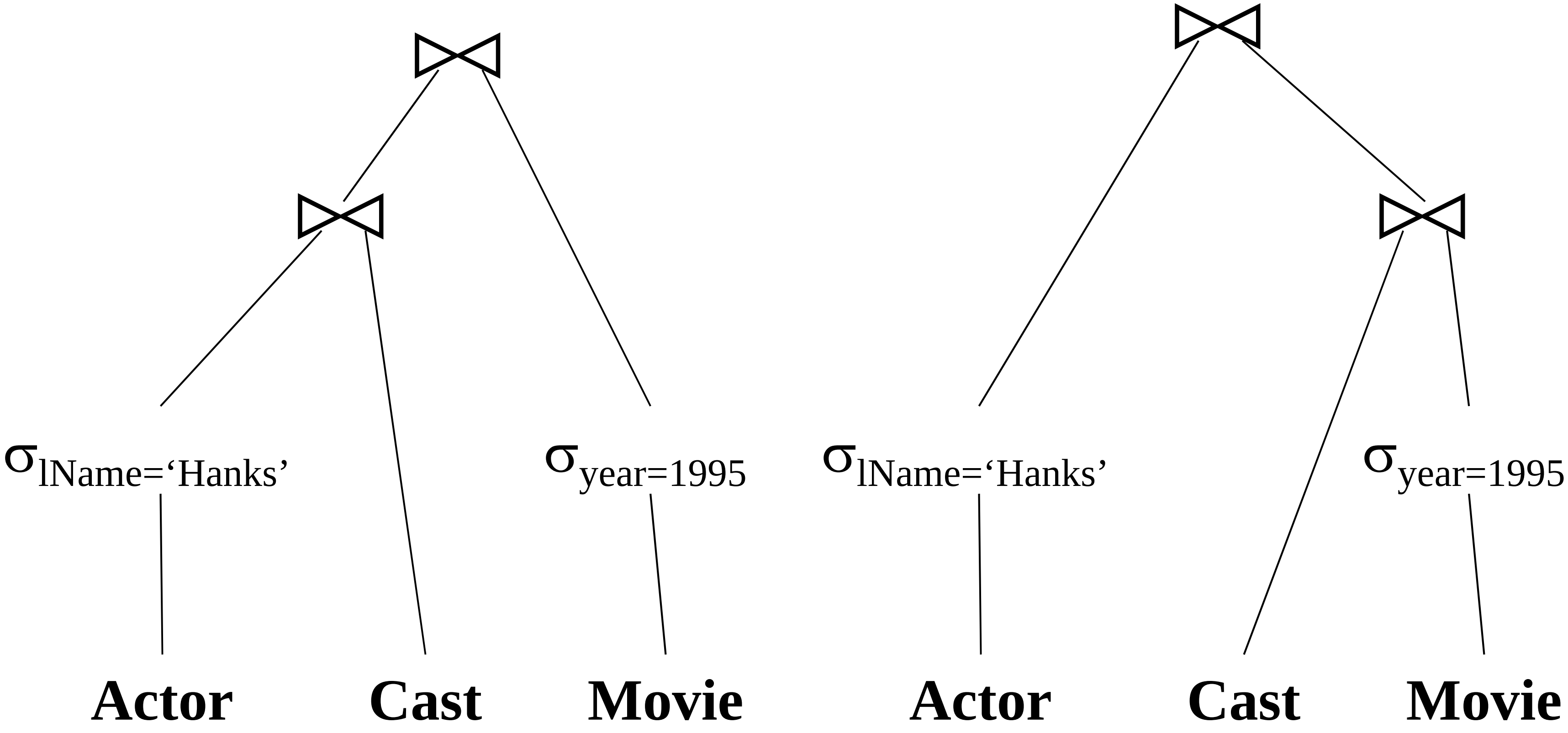
Movie:

id	Name	year
. . .		1995
. . .		

Plan 1: . . . . [in class]

Plan 2: . . . . [in class]

Evaluating Tom Hanks



Optimization and Query Execution

- Indexes: on Actor.lName, on Movie.year
- Multiple implementations of joins
- Query optimization (which join order? access path selection)
- Statistics

Now Let’s See Database Updates

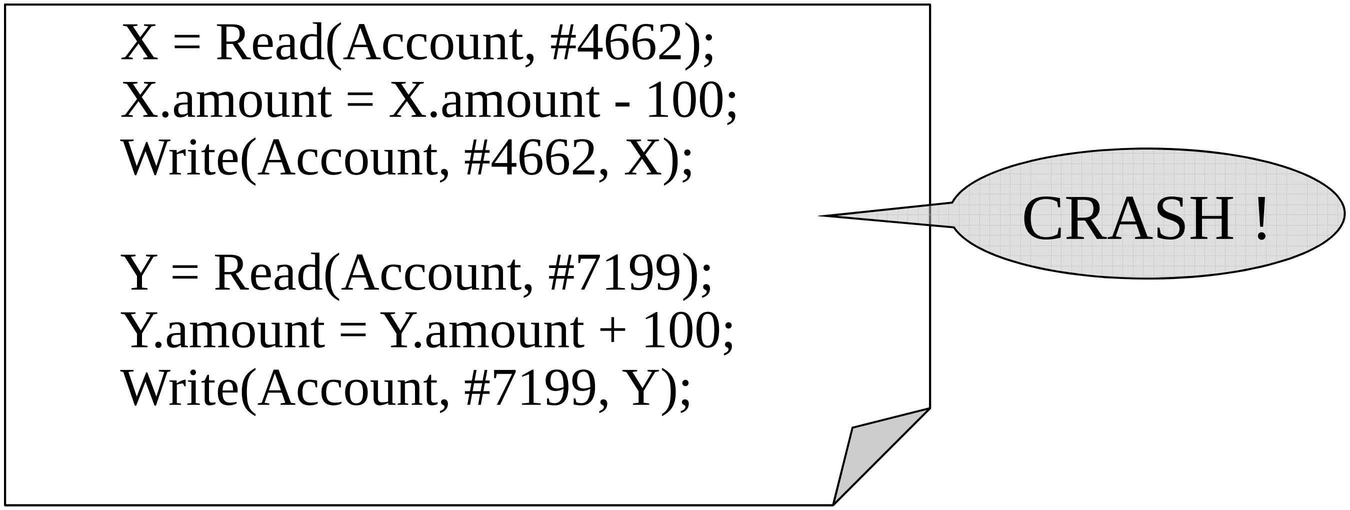
- Transfer $100 from account #4662 to #7199:

```
X = Read(Account, #4662);
X.amount = X.amount - 100;
Write(Account, #4662, X);

Y = Read(Account, #7199);
Y.amount = Y.amount + 100;
Write(Account, #7199, Y);
```

Now Let’s See Database Updates

- Transfer $100 from account #4662 to #7199:



```
X = Read(Account, #4662);
X.amount = X.amount - 100;
Write(Account, #4662, X);

Y = Read(Account, #7199);
Y.amount = Y.amount + 100;
Write(Account, #7199, Y);
```

What is the problem ?

Concurrency Control

- How to overdraft your account:

User 1

```
X = Read(Account);
if (X.amount > 100)
   { dispense_money( );
      X.amount = X.amount – 100;
    }
else error(“Insufficient funds”);
```

User 2

```
X = Read(Account);
if (X.amount > 100)
   { dispense_money( );
      X.amount = X.amount – 100;
    }
else error(“Insufficient funds”);
```

What can go wrong ?

Transactions

- Recovery
- Concurrency control

ACID =

- Atomicity (= recovery)
- Consistency
- Isolation (= concurrency control)
- Durability (= persistance)

Client/Server Architecture

- There is a single *server* that stores the database (called DBMS or RDBMS):
  - Usually a beefy system, e.g. IISQLSRV1
  - But can be your own desktop…
  - … or a huge cluster running a parallel dbms
- Many *clients* run apps and connect to DBMS
  - E.g. Microsoft's SQL Server Management Studio
  - Or psql (for postgres)
  - More realistically some Java, C#, or C++ program
- Clients "talk" to server using JDBC protocol

What This Course Contains

- SQL
- Conceptual Design
- Transactions
- Database tuning and internals (very little)
- Distributed databases: a taste of *MapReduce*
- a little XML: Xpath, Xquery

Accessing SQL Server

SQL Server Management Studio

- Server Type = Database Engine
- Server Name = IISQLSRV
- Authentication = SQL Server Authentication
  - Login = your UW netid/email address (*not* CSE email)
  - Password = [?]

Change your password !!

Then play with IMDB, start working on project 1