# Introduction to Database Systems CSE 444

Lecture 1: Introduction

# The Staff

- Instructors:
  - Wolfgang Gatterbauer (CSE 438)
  - Alexandra Meliou (CSE 444)
    - Email: cse444-instructors@cs
    - Office Hours: Wed 2pm – 3:50pm

Same as the course number, how cool is that!

- TAs:
  - Grad: Rita Sodt (rsodt@cs)
    - Office Hours: Thu 10:30am – 12pm CSE 216
  - Ugrad: Liem Dinh (liemdinh@cs)
    - Office Hours: Tue 1:30pm – 3pm CSE 220

# Communications

- Web page: http://www.cs.washington.edu/444
  - Lectures will be available there
  - The mini-projects description will be there
  - Homeworks will be posted there

- Mailing list
  - Announcements, group discussions
  - You are already subscribed

- Message board
  - Great place to ask assignment-related questions

# Textbook

- Main textbook:

  Database Systems: The Complete Book
  Hector Garcia Molina
  Jeffrey Ullman
  Jennifer Widom

- Other Texts:
  - Database Management Systems, Ramakrishnan, Gehrke
  - Foundations of Databases, Abiteboul, Hull, Vianu
  - Fundamentals of Database Systems, Elmarsi, Navathe
  - Data on the Web, Abiteboul, Buneman, Suciu

# Course Format

- Lectures: MWF, 12:30pm – 1:20pm
  - EEB 045

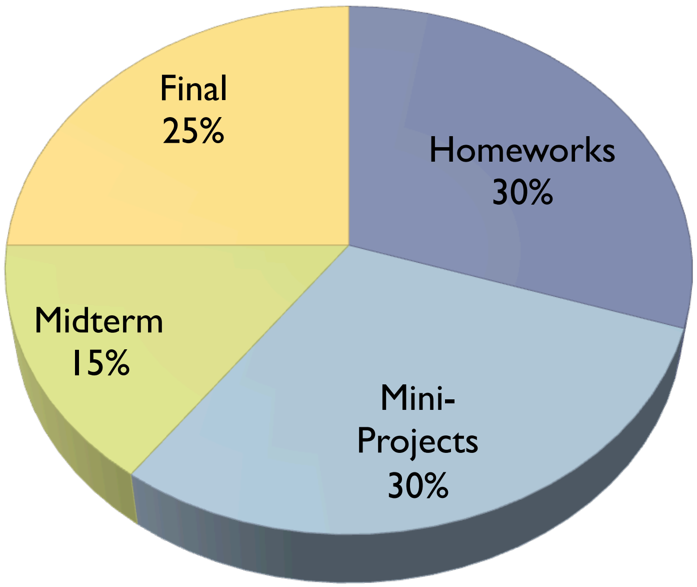
- Sections: Thu, 8:30am – 9:20am, 9:30am – 10:20am
  - EEB 025

- 4 Mini-Projects
- 3 Homework assignments

- Midterm and final exams

# Grading

| | |
|---|---|
| Homeworks | 30% |
| Mini-projects | 30% |
| Midterm | 15% |
| Final | 25% |



# Four Mini-Projects

- SQL
- SQL in Java
- Database tuning
- Parallel processing: MapReduce

Check course website for due dates

# Three Homework Assignments

- Conceptual Design
- Transactions
- Query execution and optimization

Check course website for due dates

# Exams

- Midterm: Wednesday, February 9, in class
- Final: Thursday, March 17, 8:30-10:20am, in class

# Outline of Today's Lecture

- Overview of a DBMS
- A DBMS through an example
- Course content

# Database

- What is a database ?
  - A collection of files storing related data

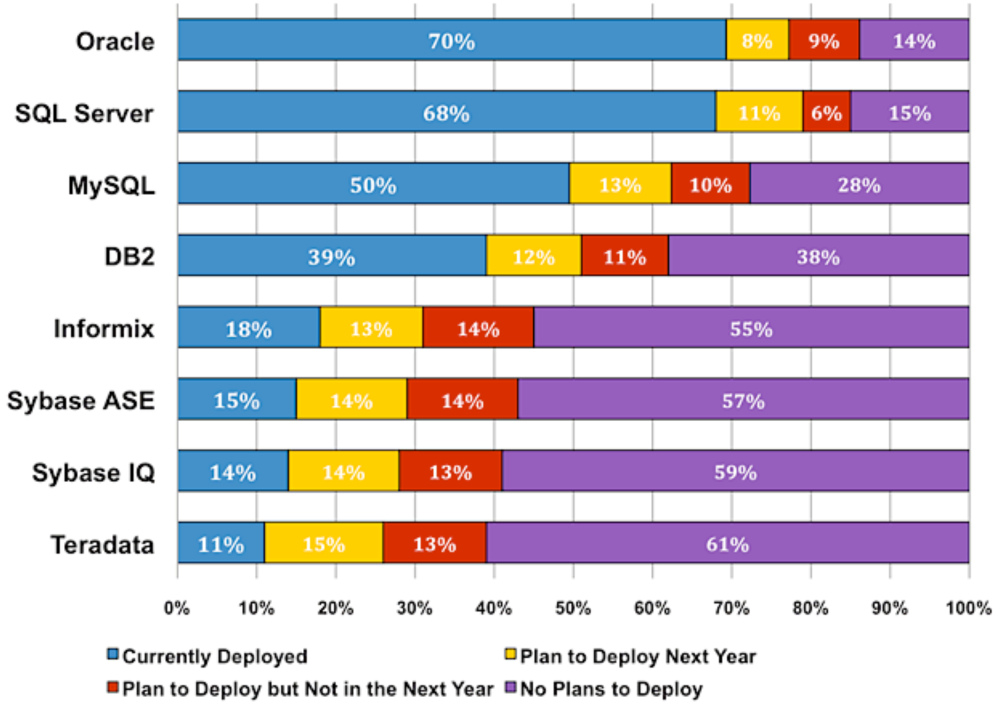
- Give examples of databases
  - Accounts database; payroll database; UW's students database; Amazon's products database; airline reservation database

# Database Management System

- What is a DBMS ?
  - A big C program written by someone else that allows us to manage efficiently a large database and allows it to persist over long periods of time
- Give examples of DBMSs
  - DB2 (IBM), SQL Server (MS), Oracle, Sybase
  - MySQL, PostgreSQL, …

# Market Shares

- From 2006 Gartner report:
  - Oracle: 47% market with $7.1BN in sales
  - IBM: 21% market with $3.2BN in sales
  - Microsoft: 17% market with $2.6BN in sales
- From 2008 Gartner study:



# 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, …

# Required Data Management Functionality

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

# DBMS Benefits

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

# Back to Example: IMDB database

**Actor**

| id | fName | lName | gender |
|---|---|---|---|

**Directors**

| id | fName | lName |
|---|---|---|

**Movie**

| id | name | year | rank |
|---|---|---|---|

**Genre**

| mid | genre |
|---|---|

**Movie_Directors**

| did | mid |
|---|---|

**Casts**

| pid | mid | role |
|---|---|---|

# Tables

**Actor:**

| id | fName | lName | gender |
|---|---|---|---|
| 429073 | Tom | Hanks | M |
| 146871 | Amy | Hanks | F |
| . . . | | | |

**Movie:**

| id | Name | year |
|---|---|---|
| 561300 | Toy Story | 1995 |
| . . . | . . . | . .. |

**Cast:**

| pid | mid | role |
|---|---|---|
| 429073 | 561300 | Woody |
| . . . | | |

# 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, 11M casts, 1.5M movies;
How long do we expect it to take?

# How Can We Evaluate the Query ?

**Actor:**

| id | fName | lName | gender |
|---|---|---|---|
| . . . |  | Hanks |  |
| . . . |  |  |  |

**Cast:**

| pid | mid |
|---|---|
| . . . |  |
| . . . |  |

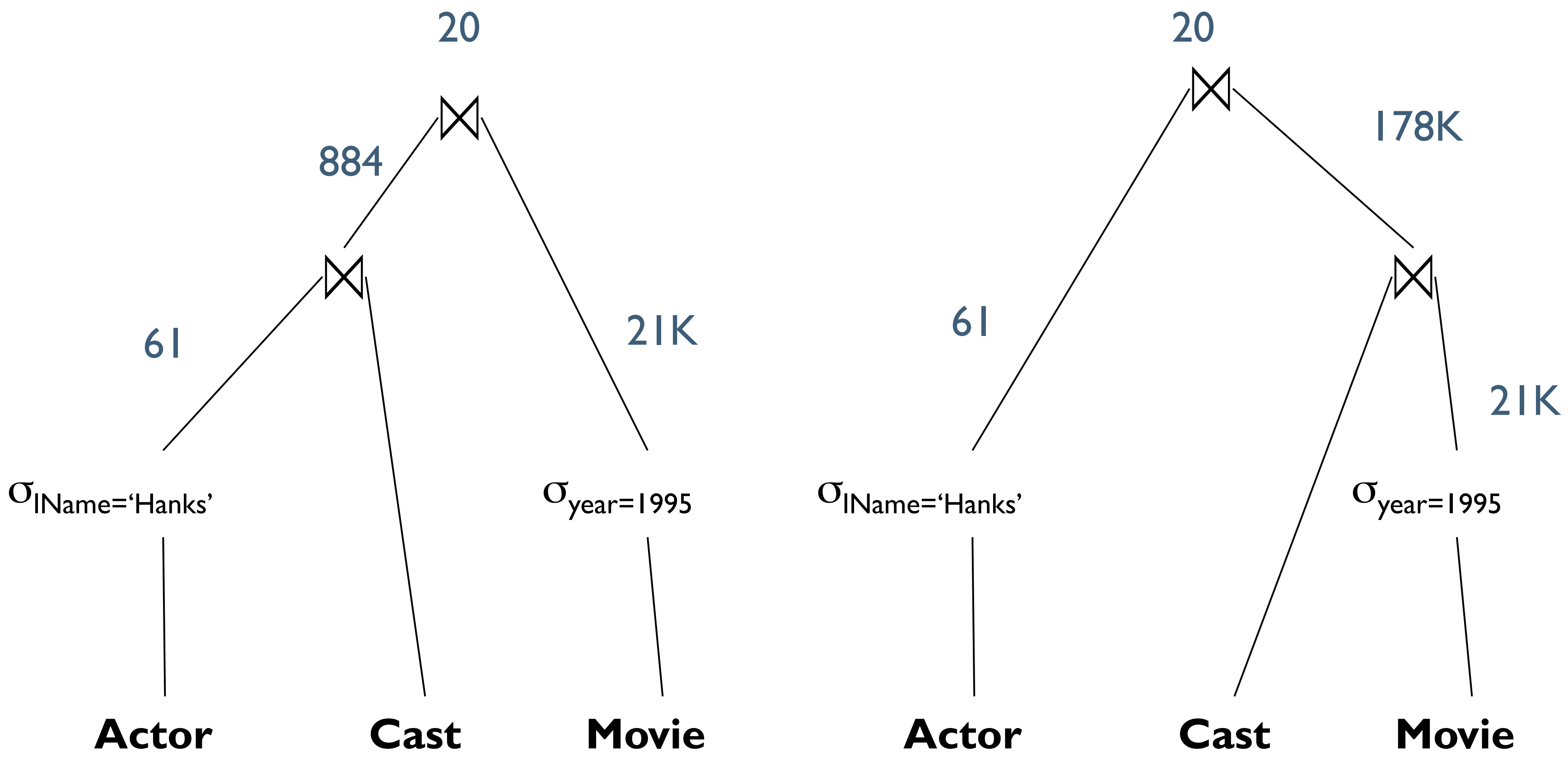
**Movie:**

| id | Name | year |
|---|---|---|
| . . . |  | 1995 |
| . . . |  |  |

Plan 1: . . . . [ in class ]

Plan 2: . . . . [ in class ]

# Evaluating Tom Hanks

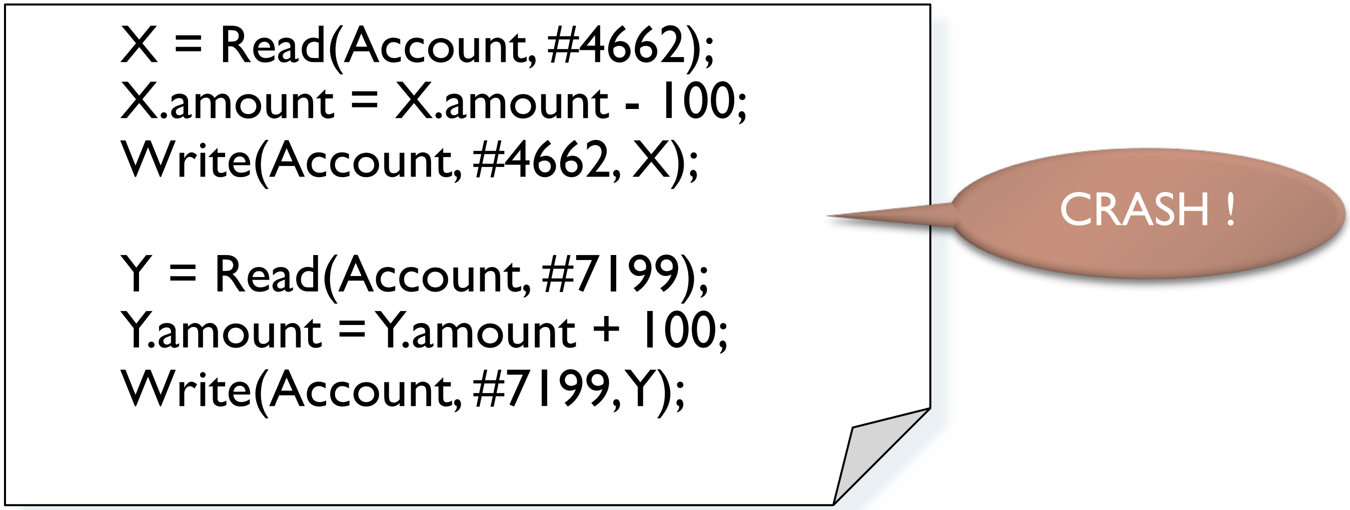


# What an RDBMS Does Well (1/2)

- Indexes: on Actor.lName, on Movie.year
- Multiple implementations of joins
- Query optimization
  - Access path selection
  - Join order
  - Join implementation
- 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);
```

CRASH !

What is the problem ?

# What a RDBMS Does Well (2/2)

- Transactions !
- Recovery
- Concurrency control

# Client/Server Architecture

- There is a single server that stores the database (called DBMS or RDBMS):
  - Usually a beefy system, e.g. IISQLSRV
  - 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 Management Studio
  - Or psql (for postgres)
  - More realistically some Java or C++ program
- Clients "talk" to server using JDBC protocol

# What This Course Contains

- SQL
- Conceptual Design
- Transactions
- Database tuning and internals (very little)
- Query Optimization
- Distributed databases: a taste of MapReduce
- More stuff depending on time