Introduction to Database Systems
CSE 444

Lecture #1
January 4, 2006

Staff

- Instructor: Dan Suciu
  - Allen, Room 662, suciu@cs.washington.edu
  - Office hours: Mondays 12:30 (or by appointment)

- TAs:
  - Fei Wu, wufei@cs.washington.edu
    - Office hours: Fridays 1:00-2:00, CSE 216
  - Nathan Bales, nbales@cs.washington.edu,
    nathan@js3k.com
Communications

• Web page:
  http://www.cs.washington.edu/444/
  – Lectures will be available here
  – Homeworks will be posted here (HW1 is posted)
  – The project description will be here

• Mailing list:
  – Announcements, group discussions
  – Please subscribe

Textbook(s)

Main textbook, available at the bookstore:

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

Most chapters are good. Some are not (functional dependencies).
COME TO CLASS ! ASK QUESTIONS ! READ SLIDES !
Other Texts

Available at the Engineering Library:

• *Database Management Systems*, Ramakrishnan
• *XQuery from the Experts*, Katz, Ed.
• *Fundamentals of Database Systems*, Elmasri, Navathe
• *Foundations of Databases*, Abiteboul, Hull, Vianu
• *Data on the Web*, Abiteboul, Buneman, Suciu

Outline of Today’s Lecture

• Overview of DBMS
• An example
• Course outline
• Assignments for Friday
Database

What is a database?

Give examples of databases

• A collection of files storing related data

• Accounts database; payroll database; UW’s students database; Amazon’s products database; airline reservation database
Database Management System

What is a DBMS?

Give examples of 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 DBMS

- DB2 (IBM), SQL Server (MS), Oracle, Sybase
- MySQL, Postgres, …
Market Shares

From 2004 www.computerworld.com

- IBM: 35% market with $2.5BN in sales
- Oracle: 33% market with $2.3BN in sales
- Microsoft: 19% market with $1.3BN in sales

An Example

The Internet Movie Database
http://www.imdb.com

- Entities:
  Actors (800k), Movies (400k), Directors, …

- Relationships:
  who played where, who directed what, …

Want to store and process locally; what functions do we need?
Functionality

1. Create/store large datasets
2. Search/query/update
3. Change the structure
4. Concurrent access to many user
5. Recover from crashes
6. Security (not here, but in other apps)

Possible Organizations

• Files
• Spreadsheets
• DBMS
1. Create/store Large Datasets

- Files: Yes, but…
- Spreadsheets: Not really…
- DBMS: Yes

2. Search/Query/Update

- Simple query:
  - In what year was ‘Rain man’ produced?
- Multi-table query:
  - Find all movies by ‘Coppola’
- Complex query:
  - For each actor, count her/his movies
- Updating:
  - Insert a new movie; add an actor to a movie; etc
2. Search/Query/Update

- Files
  - Simple queries

- Spreadsheets
  - Multi-table queries (maybe)

- DBMS
  - All

Updates: generally OK

3. Change the Structure

Add *Address* to each Actor

- Files
  - Very hard

- Spreadsheets
  - Yes

- DBMS
  - Yes
4. Concurrent Access

Multiple users access/update the data concurrently

• What can go wrong?

• How do we protect against that in OS?

• This is insufficient in databases; why?

Lost update; resulting in inconsistent data

Locks

A logical action consists of multiple updates
5. Recover from crashes

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

X = Read(Accounts, 4662);
X.amount = X.amount - 100;
Write(Accounts, 4662, X);

Y = Read(Accounts, 7199);
Y.amount = Y.amount + 100;
Write(Accounts, 7199, Y);

CRASH!

What is the problem?

Enters a DMBS

“Two tier system” or “client-server”

Data files

connection
(ODBC, JDBC)

Database server
(someone else’s
C program)

Applications
DBMS = Collection of Tables

Directors:

<table>
<thead>
<tr>
<th>id</th>
<th>fName</th>
<th>lName</th>
</tr>
</thead>
<tbody>
<tr>
<td>15901</td>
<td>Francis Ford</td>
<td>Coppola</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Movie_Directors:

<table>
<thead>
<tr>
<th>id</th>
<th>mid</th>
</tr>
</thead>
<tbody>
<tr>
<td>15901</td>
<td>130128</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Movies:

<table>
<thead>
<tr>
<th>mid</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>130128</td>
<td>The Godfather</td>
<td>1972</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Still implemented as files, but behind the scenes can be quite complex

“data independence”

1. Create/store Large Datasets

Use SQL to create and populate tables:

```sql
CREATE TABLE Actors ( Name CHAR(30), DateOfBirth CHAR(20) ) . . .
```

```sql
INSERT INTO Actors VALUES(‘Tom Hanks’, . . .)
```

Size and physical organization is handled by DBMS
We focus on modeling the database

Will study data modeling in this course
2. Searching/Querying/Updating

- Find all movies by ‘Coppola’

\[
\text{SELECT title}
\text{FROM Movies, Directors, Movie_Directors}
\text{WHERE Directors.lname = 'Coppola' and}
\text{Movies.mid = Movie_Directors.mid and}
\text{Movie_Directors.id = Directors.id}
\]

We will study SQL in gory details in this course

- What happens behind the scene ?

We will discuss the query optimizer in class.

---

Example

SQL Server Management Studio
- Server Type = Database Engine
- Server Name = IISQLSRV
- Authentication = SQL Server Authentication
  - Login = your login
  - Password = [your student id]!A

Change your password !!

Then play with IMDB, create your own DB
3. Changing the Structure

Add \textit{Address} to each Actor

\begin{verbatim}
ALTER TABLE Actor
ADD address CHAR(50)
DEFAULT 'unknown'
\end{verbatim}

Lots of cleverness goes on behind the scenes

3&4 Concurrency&Recovery: Transactions

- A \textit{transaction} = sequence of statements that either all succeed, or all fail
- E.g. Transfer $100

\begin{verbatim}
BEGIN TRANSACTION;
UPDATE Accounts
SET amount = amount - 100
WHERE number = 4662
UPDATE Accounts
SET amount = amount + 100
WHERE number = 7199
COMMIT
\end{verbatim}
Transactions

- Transactions have the ACID properties:
  A = atomicity
  C = consistency
  I = isolation
  D = durability

4. Concurrent Access

- Serializable execution of transactions
  - The I (=isolation) in ACID

We study three techniques in this course

- Locks
- Timestamps
- Validation
5. Recovery from crashes

• Every transaction either executes completely, or doesn’t execute at all
  – The A (=atomicity) in ACID

We study three types of log files in this course

Undo log file
Redo log file
Undo/Redo log file

Course Outline

Part I
• SQL, Relational model, database design
• XML, XPath, XQuery
• Database security, Transactions
Midterm: Friday, February 10th (in class)

Part II
• Concurrency control and recovery
• Query execution and optimization
Final: March 15th, 2:30-4:20 (this room)
Structure

• Prerequisites: Data structures course (CSE-326).

• Work & Grading:
  – Homework: 25% (4 of them; some light programming)
  – Project: 30% (next)
  – Midterm: 15%
  – Final: 25%
  – Intangibles: 5%

The Project

• Models data management needs of a company

• Will have two phases
  – Correspond to Real World phases of system evolution in a company

• We use SQL Server, C#, .NET
  – You may use other technologies, but we offer no support

• More details soon
Final Announcements

• Reading assignment for Friday:
  – **Introduction** from *SQL for Web Nerds*,

• Login SQL Server
  – User name = your U email address
  – Password = "studentID" + "!A"

• Homework 1 is posted on the Web: due on January 18