Introduction to Database Systems
CSE 444

Lecture #1
September 30, 2002

1

Staff

- Instructor: Dan Suciu
  - Sieg, Room 318, suciu@cs.washington.edu
  - Office hours: Monday, 11:30-12:30
  - (or by appointment)
- TA: Yana Kadiyska
  - yana@cs.washington.edu
  - Office hours: TBA (check mailing list)

2

Communications

- Web page:
  http://www.cs.washington.edu/444/
- Mailing list: send email to
  majordomo@cs
  saying:
  subscribe cse444

3

Textbook(s)

Main textbook, available at the bookstore:

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

Almost identical, and also available at the bookstore:

- *A First Course in Database Systems*, Jeff Ullman and Jennifer Widom
- *Database Implementation*, Hector Garcia-Molina, Jeff Ullman and Jennifer Widom

4

Other Texts

On reserve at the Engineering Library:

- *Database Management Systems*, Ramakrishnan
  - very comprehensive
- *Fundamentals of Database Systems*, Elmasri, Navathe
  - very widely used
- *Foundations of Databases*, Abiteboul, Hull, Vianu
  - Mostly theory of databases
- *Data on the Web*, Abiteboul, Buneman, Suciu
  - XML and other new/advanced stuff

5

Other Required Readings

There will be reading assignments from the Web:

- **SQL for Web Nerds,** by Philip Greenspun, http://philip.greenspun.com/sql/
- Others, especially for XML

For SQL, a good source of information is the MSDN library (on your Windows machine)

6

Outline for Today's Lecture

- Overview of database systems
  - Reading assignment for next lecture (Wednesday): from **SQL for Web Nerds,** by Philip Greenspun, Introduction http://philip.greenspun.com/sql/
- Course Outline
- Structure of the course

7

What *Is* a Relational Database Management System ?

Database Management System = DBMS

Relational DBMS = RDBMS

- A collection of files that store the data
- A big C program written by someone else that accesses and updates those files for you

8

Where are RDBMS used ?

- Backend for traditional "database" applications
- Backend for large Websites
- Backend for Web services

9

Example of a Traditional Database Application

Suppose we are building a system to store the information about:

- students
- courses
- professors
- who takes what, who teaches what

10

Can we do it without a DBMS ?

Sure we can! Start by storing the data in files:

students.txt courses.txt professors.txt

Now write C or Java programs to implement specific tasks

11

Doing it without a DBMS...

- Enroll "Mary Johnson" in "CSE444":

Write a C program to do the following:

Read 'students.txt'
Read 'courses.txt'
Find&update the record "Mary Johnson"
Find&update the record "CSE444"
Write "students.txt"
Write "courses.txt"

12

Problems without an DBMS...

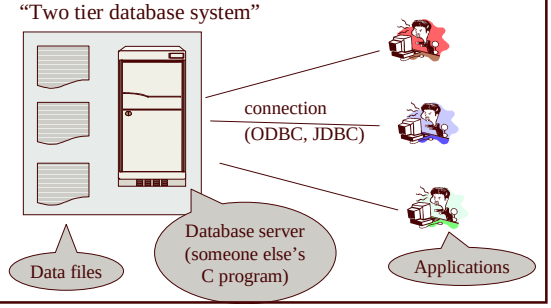
- System crashes:

  - What is the problem ?
- Large data sets (say 50GB)
  - What is the problem ?
- Simultaneous access by many users
  - Need locks: we know them from OS, but now data on disk; and is there any fun to re-implement them ?

13

Enters a DMBS

"Two tier database system"



Functionality of a DBMS

The programmer sees SQL, which has two components:

- Data Definition Language - DDL
- Data Manipulation Language - DML
  - query language

Behind the scenes the DBMS has:

- Query optimizer
- Query engine
- Storage management
- Transaction Management (concurrency, recovery)

15

Functionality of a DBMS

Two things to remember:

- Client-server architecture
  - Slow, cumbersome connection
  - But good for the data
- It is just someone else's C program
  - In the beginning we may be impressed by its speed
  - But later we discover that it can be frustratingly slow
  - We can do any particular task faster outside the DBMS
  - But the DBMS is *general* and *convenient*

16

How the Programmer Sees the DBMS

- Start with DDL to *create tables*:

```
CREATE TABLE Students (
        Name CHAR(30)
        SSN CHAR(9) PRIMARY KEY NOT NULL,
        Category CHAR(20)
) . . .
```

- Continue with DML to *populate tables:*

```
INSERT INTO Students
VALUES('Charles', '123456789', 'undergraduate')
. . . .
```

17

How the Programmer Sees the DBMS

- Tables:

Students:

SSN	Name	Category
123-45-6789	Charles	undergrad
234-56-7890	Dan	grad
	...	...

Takes:

SSN	CID
123-45-6789	CSE444
123-45-6789	CSE444
234-56-7890	CSE142
	...

Courses:

CID	Name	Quarter
CSE444	Databases	fall
CSE541	Operating systems	winter

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

  "*data independence*" = separate *logical* view from *physical implementation*

18

Transactions

- Enroll "Mary Johnson" in "CSE444":

```
BEGIN TRANSACTION;

INSERT INTO Takes
   SELECT Students.SSN, Courses.CID
   FROM Students, Courses
   WHERE Students.name = 'Mary Johnson' and
            Courses.name = 'CSE444'

-- More updates here....

IF everything-went-OK
     THEN COMMIT;
ELSE ROLLBACK
```

If system crashes, the transaction is still either committed or aborted

Transactions

- A *transaction* = sequence of statements that either all succeed, or all fail
- Transactions have the ACID properties:
  - A = atomicity
  - C = consistency
  - I = independence
  - D = durability

Queries

- Find all courses that "Mary" takes

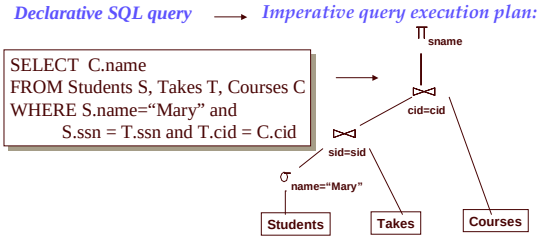
```
SELECT  C.name
FROM    Students S, Takes T, Courses C
WHERE   S.name="Mary" and
        S.ssn = T.ssn and T.cid = C.cid
```

- What happens behind the scene ?
  - Query processor figures out how to answer the query efficiently.

Queries, behind the scene

Declarative SQL query → *Imperative query execution plan:*

```
SELECT  C.name
FROM Students S, Takes T, Courses C
WHERE S.name="Mary" and
      S.ssn = T.ssn and T.cid = C.cid
```



The **optimizer** chooses the best execution plan for a query

Database Systems

- The big commercial database vendors:
  - Oracle
  - IBM (with DB2) bought Informix recently
  - Microsoft (SQL Server)
  - Sybase
- Some free database systems (Unix) :
  - Postgres
  - Mysql
  - Predator
- In CSE444 we use SQL Server. You may use something else, but you are on your own.

New Trends in Databases

- Object-relational databases
- Main memory database systems
- XML XML XML !
  - Relational databases with XML support
  - Middleware between XML and relational databases
  - Native XML database systems
  - Lots of research here at UW on XML and databases
- Peer to peer, stream data management – still research

Course Outline (may vary slightly)

Part I
- SQL (Chapter 7)
- The relational data model (Chapter 3)
- Database design (Chapters 2, 3, 7)
- XML, XPath, XQuery

Midterm: November 1st

Part II
- Data storage, indexes (Chapters 11-13)
- Query execution and optimization (Chapter 15,16)
- Recovery (Chapter 17)

Final: December 13th

25

Structure

- Prerequisites: Data structures course (CSE-326 or equivalent).
- Work & Grading:
  - Homework 25%: 6 of them, some light programming.
  - Project: 25% - see next.
  - Midterm: 20%
  - Final: 25%
  - Intangibles: 5%

26

The Project

- Goal: design end-to-end database application.
- Work in groups of 3-4 (start forming ***now***).
- Topic: design a multi-user calendar:
  - Store the data in a DBMS (SQL Server)
  - Implement a Web interface to it
  - Implement a Webservice over it

27

The Project

- Grading based on:
  - Functionality (the more the better) (say 80%)
  - Implementation, efficiency (say 20%)
- There will be some milestones to turn in during the quarter
  - We want to make sure that you make progress
  - Do not necessarily expect feedback: ask, if you need feedback

28

The Project

Alternative topics:
- You may choose any different topic; e.g. from here:
  - http://abstract.cs.washington.edu/~zahorjan/481-02au/cse-access/overview.cgi
- It needs to include all three components:
  - A Database
  - A Website
  - A Webservice
- You need to write a 1-2 page proposal and turn it in
- But you are at your own risk (i.e. we offer little support, and grading may be less predicatble)

29

So what is this course about, really ?

- SQL:
  - An old language, but still cute
- Newer, XML stuff
  - Unfortunately less programming here
- Theory !
- Lots of implementation and hacking !
  - And you need to learn a lot while you go

30