Outline for Today’s Lecture

• Overview of database systems
• Course Outline
• First Steps in SQL
Staff

• Lecturer:
  – Heinz Stockinger
    • Heinz.Stockinger@isb-sib.ch
  – Office hours: by appointment

• TAs:
  – Gleb Skobeltsyn,
  – Wojciech Galuba
  – Šarūnas Girdzijauskas
  – Ali Salehi

Contact details

• Lecturer:
  – Heinz.Stockinger@isb-sib.ch
    • Swiss Institute of Bioinformatics
  – Batiment Genopode, office 2016 (UNIL campus)
  – Personal Web site: http://cern.ch/hst
Communications

- Web page: lsirwww.epfl.ch:
  - http://lsirwww.epfl.ch/courses/iis/2007ss
  - Lecture slides will be available here
  - Homework and solutions will be posted here
  - The project description and resources will be here

- Newsgroup:
  - epfl.ic.cours.IIS

Main Textbook

- *Databases and Transaction Processing*,
  *An application-oriented approach*
Other Texts

Many classic textbooks (each of them will do it)
• **Database Systems: The Complete Book**, Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom
• **Database Management Systems**, Ramakrishnan
• **Fundamentals of Database Systems**, Elmasri, Navathe
• **Database Systems**, Date (7th edition)
• **Modern Database Management**, Hoffer, (4th edition)

Material on the Web

SQL Introduction

Java Technology:
– java.sun.com

Web Technology
– www.w3c.org (Specifications/standards)
The Course

• Goal: Teaching
  – relational database management system (RDBMS) (standard)
  – with a strong emphasis on the Web

• Fortunately, others already did it already
  – Alon Halevy, Dan Suciu, Univ. of Washington
  – http://www.cs.washington.edu/education/courses/cse444/
  – Lecture was even awarded a price!

Acknowledgement

• Build on UoW course
  – many slides
  – many exercise
  – ideas for the project

• Main difference
  – less theory
  – will use real Web data in the project

• Prof. Aberer previously taught this course in Summer Term 2005
Let’s get started with databases

What is behind this Web Site?

• http://immo.search.ch/
• Search on a large database
• Specify search conditions
• Many users
• Updates
• Access through a web interface
Database Management Systems

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

Relational DBMS = RDBMS
- Data files are structured as relations (tables)
Where are RDBMS used?

- **Backend** for traditional “database” applications
  - EPFL administration
- Backend for large Websites
  - Immosearch
- Backend for Web services
  - Amazon

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
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.

---

Doing it *without* a DBMS...

- Enroll “Mary Johnson” in “CSE444”:

  Write a C++/Java 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”

---
Problems *without* an DBMS...

- System crashes:
  - What is the problem?
- Large data sets (say 50GB)
  - Why is this a problem?
- Simultaneous access by many users
  - Lock students.txt – what is the problem?

Using a DBMS

“Two tier system” or “client-server”

- Data files
- Database server (someone else’s C/C++ program)
- Applications
- Connection (ODBC, JDBC)
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 engine
- Query optimizer
- Storage management
- Transaction Management (concurrency, recovery)

How the Programmer Sees the DBMS - 1

- Start with DDL to *create tables*:
  ```sql
  CREATE TABLE Students (   
    Name CHAR(30)   
    SSN CHAR(9) PRIMARY KEY NOT NULL,   
    Category CHAR(20)   
  ) . . .
  ```

- Continue with DML to *populate tables*:
  ```sql
  INSERT INTO Students VALUES(‘Charles’, ‘123456789’, ‘undergraduate’)   
  . . .
  ```
How the Programmer Sees the DBMS - 2

- Tables:
  Students:
  - SSN: 123-45-6789, Name: Charles, Category: undergrad
  - SSN: 234-56-7890, Name: Dan, Category: grad

  Takes:
  - SSN: 123-45-6789, CID: CSE444
  - SSN: 234-56-7890, CID: CSE142

  Courses:
  - CID: CSE444, Name: Databases, Quarter: fall
  - CID: CSE541, Name: Operating systems, Quarter: winter

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

  "data independence" = separate logical view from physical implementation

Queries

- Find all courses that "Mary" takes

  ```sql
  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.

Transactions - 1

- 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 - 2

• A transaction = sequence of statements that either all succeed, or all fail
• Transactions have the ACID properties:
  A = atomicity (a transaction should be done or undone completely)
  C = consistency (a transaction should transform a system from one consistent state to another consistent state)
  I = isolation (each transaction should happen independently of other transactions)
  D = durability (completed transactions should remain permanent)

Database Systems

• The big commercial database vendors:
  – Oracle
  – IBM (with DB2)
  – Microsoft (SQL Server)
  – Sybase
• Some free database systems (UNIX) :
  – Postgres
  – MySQL
  – Predator
Databases and the Web

• Accessing databases through web interfaces
  – Java programming interface (JDBC)
  – Embedding into HTML pages (JSP)
  – Access through HTTP protocol (Web Services)
• Using Web document formats for data definition and manipulation
  – XML, XQuery, XPath
  – XML databases and messaging systems

Database Integration

• Combining data from different databases
  – collection of data (wrapping)
  – combination of data and generation of new views on the data (mediation)
• Problem: heterogeneity
  – access, representation, content
• Example revisited
  – http://immo.search.ch/
  – http://www.swissimmo.ch
Other Trends in Databases

• Industrial
  – Object-relational databases
  – Main memory database systems
  – Data warehousing and mining

• Research
  – Peer-to-peer data management
  – Stream data management
  – Mobile data management

Back to the general overview of course
Structure

• Prerequisites:
  – Programming courses (mainly Java)
  – Data structures

• Work & Grading:
  – Homework (4): 0%
  – Exam (similar to homework): 50%
  – Project: 50% (see next)
    • each phase graded separately
    • includes discussion

The Project

• Models the real data management needs of a Web company
  – Phase 1: Modelling and Data Acquisition
  – Phase 2: Simple Web Service
  – Phase 3: Data integration and Web service interaction

• "One can only start to appreciate database systems by actually trying to use one" (Halevy)

• Any SW/IT company will love you for these skills 😊
The Project – Side Effects

• Trains your **soft skills**
  – team work
  – deal with bugs, poor documentation, …
  – produce with limited time resources
  – project management and reporting
• Results useful for you personally
  – Demo
  – Project should be fun 😊

Practical Concerns

• Project is rather work intensive
• Important to keep time schedule
• Communication through Web
• Newsgroup
So what is this course about, really?

A bit of everything!

- Languages: SQL, XPath, Xquery
- Data modeling
- Theory! (Functional dependencies, normal forms)
- Web services
- Algorithms and data structures (in the second half)
- Lots of implementation for the project
- Most importantly: how to meet Real World needs
Summary

• We use a (Relational) Database Management System:
  – Mainly as the backend
  – To store different kinds of data
  – To allow for concurrent access of many users
  – To ensure that data is not corrupted