Introduction to Database Systems

CS4320

Instructor: Christoph Koch
koch@cs.cornell.edu
CS4320/1: Introduction to Database Systems

Underlying theme: How do I build a data management system?

• CS4320 will deal with the underlying concepts
  • No programming assignments
• CS4322 will be the practicum
  • Build components of a small search engine (C++ programming)
Information is one of the most valuable resources in this information age.

- How do we effectively and efficiently manage this information?
- Relational database management systems
  - Dominant data management paradigm today
- Search engines
  - Ubiquitous today
- 100+ billion dollar a year industry
  - You will see this in the job market!
## RDBMS Market

**Worldwide 2005 Vendor Revenue Estimates from RDBMS Software, Based on Total Software Revenue (Millions of Dollars)**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>6,721.1</td>
<td>48.6</td>
<td>6,234.1</td>
<td>48.9</td>
<td>7.8</td>
</tr>
<tr>
<td>IBM</td>
<td>3,040.7</td>
<td>22.0</td>
<td>2,860.4</td>
<td>22.4</td>
<td>6.3</td>
</tr>
<tr>
<td>Microsoft</td>
<td>2,073.2</td>
<td>15.0</td>
<td>1,777.9</td>
<td>13.9</td>
<td>16.6</td>
</tr>
<tr>
<td>Teradata</td>
<td>440.7</td>
<td>3.2</td>
<td>412.1</td>
<td>3.2</td>
<td>6.9</td>
</tr>
<tr>
<td>Sybase</td>
<td>407.0</td>
<td>2.9</td>
<td>382.8</td>
<td>3.0</td>
<td>6.3</td>
</tr>
<tr>
<td>Other Vendors</td>
<td>1,134.7</td>
<td>8.2</td>
<td>1,090.4</td>
<td>8.5</td>
<td>4.1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>13,817.4</strong></td>
<td><strong>100.0</strong></td>
<td><strong>12,757.8</strong></td>
<td><strong>100.0</strong></td>
<td><strong>8.3</strong></td>
</tr>
</tbody>
</table>

Source: Gartner Dataquest (May 2006)
World's largest software companies

From Wikipedia, the free encyclopedia

The Forbes Global 2000 includes the following list of the world's largest software companies.

<table>
<thead>
<tr>
<th>Relative rank</th>
<th>Global rank</th>
<th>Name</th>
<th>Country</th>
<th>Sales ($bil)</th>
<th>Profits ($bil)</th>
<th>Assets ($bil)</th>
<th>Market Value ($bil)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54</td>
<td>Microsoft</td>
<td>United States</td>
<td>41.36</td>
<td>13.06</td>
<td>67.26</td>
<td>279.02</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>Oracle</td>
<td>United States</td>
<td>12.89</td>
<td>2.88</td>
<td>19.35</td>
<td>64.01</td>
</tr>
<tr>
<td>3</td>
<td>248</td>
<td>First Data</td>
<td>United States</td>
<td>10.49</td>
<td>1.59</td>
<td>34.25</td>
<td>34.43</td>
</tr>
<tr>
<td>4</td>
<td>381</td>
<td>SAP</td>
<td>Germany</td>
<td>10.06</td>
<td>1.77</td>
<td>10.43</td>
<td>63.10</td>
</tr>
<tr>
<td>5</td>
<td>418</td>
<td>Accenture</td>
<td>Bermuda</td>
<td>17.57</td>
<td>0.96</td>
<td>8.12</td>
<td>27.77</td>
</tr>
<tr>
<td>6</td>
<td>439</td>
<td>Google</td>
<td>United States</td>
<td>6.14</td>
<td>1.47</td>
<td>10.27</td>
<td>107.17</td>
</tr>
<tr>
<td>7</td>
<td>473</td>
<td>Yahoo</td>
<td>United States</td>
<td>5.26</td>
<td>1.90</td>
<td>10.87</td>
<td>45.48</td>
</tr>
<tr>
<td>8</td>
<td>499</td>
<td>Computer Sciences Corporation</td>
<td>United States</td>
<td>14.61</td>
<td>0.85</td>
<td>12.62</td>
<td>10.12</td>
</tr>
<tr>
<td>9</td>
<td>715</td>
<td>Electronic Data Systems</td>
<td>United States</td>
<td>19.76</td>
<td>0.15</td>
<td>17.09</td>
<td>13.84</td>
</tr>
<tr>
<td>10</td>
<td>776</td>
<td>SoftBank</td>
<td>Japan</td>
<td>7.81</td>
<td>-0.56</td>
<td>15.53</td>
<td>32.78</td>
</tr>
<tr>
<td>11</td>
<td>948</td>
<td>Symantec</td>
<td>United States</td>
<td>3.62</td>
<td>0.16</td>
<td>17.68</td>
<td>17.60</td>
</tr>
<tr>
<td>12</td>
<td>959</td>
<td>CA</td>
<td>United States</td>
<td>3.76</td>
<td>0.22</td>
<td>10.06</td>
<td>15.73</td>
</tr>
<tr>
<td>13</td>
<td>993</td>
<td>Fiserv</td>
<td>United States</td>
<td>4.06</td>
<td>0.52</td>
<td>6.04</td>
<td>7.70</td>
</tr>
<tr>
<td>14</td>
<td>1010</td>
<td>Affiliated Computer Services</td>
<td>United States</td>
<td>4.94</td>
<td>0.42</td>
<td>5.31</td>
<td>7.86</td>
</tr>
<tr>
<td>15</td>
<td>1077</td>
<td>Adobe Systems</td>
<td>United States</td>
<td>1.97</td>
<td>0.60</td>
<td>2.44</td>
<td>23.12</td>
</tr>
<tr>
<td>16</td>
<td>1086</td>
<td>Capgemini Group</td>
<td>France</td>
<td>8.22</td>
<td>0.17</td>
<td>7.15</td>
<td>6.50</td>
</tr>
</tbody>
</table>

„Note: The above list does not include companies like IBM whose software/services part is bigger than Microsoft. In the Forbes2000 report IBM and HP were listed as Technology Hardware companies.“
# YEAR IN REVIEW

## RESULTS OF CONTINUING OPERATIONS

### Revenue

(Dollars in millions)

<table>
<thead>
<tr>
<th></th>
<th>2006</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FOR THE YEAR ENDED DECEMBER 31:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Statement of Earnings Revenue Presentation:</strong></td>
<td></td>
</tr>
<tr>
<td>Global Services</td>
<td>$48,247</td>
</tr>
<tr>
<td>Hardware</td>
<td>22,499</td>
</tr>
<tr>
<td>Software</td>
<td>18,204</td>
</tr>
<tr>
<td>Global Financing</td>
<td>2,379</td>
</tr>
<tr>
<td>Other</td>
<td>94</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$91,424</td>
</tr>
</tbody>
</table>
Sergey Brin's Home Page

Ph.D. student in Computer Science at Stanford -
sergey@cs.stanford.edu

Research

Currently I am at Google.

In fall '98 I taught CS 349.

Data Mining

A major research interest is data mining and I run a meeting group here at Stanford. For more information take a look at the MIDAS home page or see the datamine mailing list archive. Here are some recent publications:
CS4320 Prerequisites

Courses

• CS212 (Computers and Programming)
• CS312 (Structure and Interpretation of Computer Programs)
People

• Instructor:
  • Christoph Koch

• TAs:
  • Shuang Zhao
  • Guozhang Wang
Access to Instructor and TAs

- Office hours
  - Posted on course web site
    http://www.cs.cornell.edu/courses/cs4320/

- TA mailing list
  - cs4320ta-l@cs.cornell.edu
  - cs4321ta-l@cs.cornell.edu

- Do not directly email TAs
Course Structure

• Two components
  • Assignments (50%)
    • Five assignments
    • Each assignment worth 10% of total grade.
  • Two examinations (50%)
• No programming assignments in CS4320
  • CS4321 will have all programming assignments
Textbook

• Textbook: “Database Management Systems” (3rd Edition)
  • By R. Ramakrishnan and J. Gehrke
  • Required textbook

• Syllabus
  • Defined by class lectures
  • Not defined by textbook
Assignment Policies

- Assignments have to be done individually
  - No collaboration with others
- Academic integrity violations taken VERY seriously
  - Read Cornell and CS academic integrity policies
  - Available off course web page
  - Need to sign and hand in form
- Course management system used to post assignment grades
Assignment Policies (ctd.)

- Late submissions
  - Grace period of 48 hours during which you can still achieve 90% of the full score.
  - After that: 0% of grade for even later submissions
  - No exceptions (assignments handed out well in advance of deadline)

- Regrade requests
  - Within 7 days after assignments are graded
  - Hard deadline
Exams

• Mid-term exam (20%)
  • 23 October 2008, 7:30-10:00pm
  • Closed book exam

• Final exam (30%)
  • Date TBA
  • Closed book exam
  • Cumulative with emphasis on second half

• Do not schedule other exams or interviews on these days
Relationship to CS4321

- CS4320 is about *concepts* underlying databases
  - No programming assignments
- CS4321 is the *practicum* associated with CS4320
  - Will actually build the core of a “realistic” database management system.
  - C++ programming
- Complementary
  - Suggest that you take both
  - **Can** take CS4320 without taking CS4321
  - **Cannot** take CS4321 without taking CS4320
Is CS4320/4321 a lot of work?

• It depends!
  • Much of the material in CS4320 is probably new to you
  • CS4321 has substantial programming assignments

• Then why on earth should I take this course?
  • Intellectual argument
    • Big conceptual ideas
    • Meeting of theory and practice
  • Utilitarian argument
    • Many, many real applications (data management, data-driven websites, search engines,…)
    • Job market!
Reminder

- Complete academic integrity form (download from course homepage)
- Hand in this week in class!
What Is a DBMS?

- A very large, integrated collection of data.
- Models real-world enterprise.
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS564)
- A Database Management System (DBMS) is a software package designed to store and manage databases.
Files vs. DBMS

- Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control
Why Use a DBMS?

• Data independence and efficient access.
• Reduced application development time.
• Data integrity and security.
• Uniform data administration.
• Concurrent access, recovery from crashes.
Why Study Databases??

- Shift from *computation* to *information*
  - at the “low end”: scramble to webspace (a mess!)
  - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project
  - ... need for DBMS exploding
- DBMS encompasses most of CS
  - OS, languages, theory, AI, multimedia, logic
Data Models

• A **data model** is a collection of concepts for describing data.

• A **schema** is a description of a particular collection of data, using the given data model.

• The **relational model of data** is the most widely used model today.
  • Main concept: **relation**, basically a table with rows and columns.
  • Every relation has a **schema**, which describes the columns, or fields.
Levels of Abstraction

- Many **views**, single conceptual (logical) schema and **physical schema**.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.

* Schemas are defined using DDL; data is modified/queried using DML.
Example: University Database

• Conceptual schema:
  • Students(sid: string, name: string, login: string, age: integer, gpa:real)
  • Courses(cid: string, cname:string, credits:integer)
  • Enrolled(sid:string, cid:string, grade:string)

• Physical schema:
  • Relations stored as unordered files.
  • Index on first column of Students.

• External Schema (View):
  • Course_info(cid:string, enrollment:integer)
Data Independence *

- Applications insulated from how data is structured and stored.
- **Logical data independence**: Protection from changes in logical structure of data.
- **Physical data independence**: Protection from changes in physical structure of data.

* One of the most important benefits of using a DBMS!
Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.
- DBMS ensures such problems don’t arise: users can pretend they are using a single-user system.
Transaction: An Execution of a DB Program

• Key concept is transaction, which is an atomic sequence of database actions (reads/writes).
• Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
  • Users can specify some simple integrity constraints on the data, and the DBMS will enforce these constraints.
  • Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
  • Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user’s responsibility!
Scheduling Concurrent Transactions

• DBMS ensures that execution of \{T_1, \ldots, T_n\} is equivalent to some \textit{serial} execution \(T_1' \ldots T_n'\).
  • Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (\textit{Strict 2PL locking protocol}.)
  • Idea: If an action of \(T_i\) (say, writing \(X\)) affects \(T_j\) (which perhaps reads \(X\)), one of them, say \(T_i\), will obtain the lock on \(X\) first and \(T_j\) is forced to wait until \(T_i\) completes; this effectively orders the transactions.
  • What if \(T_j\) already has a lock on \(Y\) and \(T_i\) later requests a lock on \(Y\)? (\textit{Deadlock}!) \(T_i\) or \(T_j\) is \textit{aborted} and restarted!
Ensuring Atomicity

- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.

- **Idea:** Keep a *log* (history) of all actions carried out by the DBMS while executing a set of Xacts:
  - **Before** a change is made to the database, the corresponding log entry is forced to a safe location. (*WAL protocol*; OS support for this is often inadequate.)
  - **After** a crash, the effects of partially executed transactions are *undone* using the log. (Thanks to WAL, if log entry wasn’t saved before the crash, corresponding change was not applied to database!)
The Log

- The following actions are recorded in the log:
  - *Ti writes an object:* The old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *Ti commits/aborts:* A log record indicating this action.

- Log records chained together by Xact id, so it’s easy to undo a specific Xact (e.g., to resolve a deadlock).

- Log is often *duplexed* and *archived* on “stable” storage.

- All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.
Databases make these folks happy ...

- End users and DBMS vendors
- DB application programmers
  - E.g., smart webmasters
- Database administrator (DBA)
  - Designs logical / physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

Must understand how a DBMS works!
Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.

These layers must consider concurrency control and recovery.

- Query Optimization and Execution
- Relational Operators
- Files and Access Methods
- Buffer Management
- Disk Space Management

DB
Summary

• DBMS used to maintain, query large datasets.
• Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
• Levels of abstraction give data independence.
• A DBMS typically has a layered architecture.
• DBAs hold responsible jobs and are well-paid! ☺
• DBMS R&D is one of the broadest, most exciting areas in CS.