CS2Bh: Current Technologies

Introduction to XML and Relational Databases

Spring 2005

Introduction to Databases

Why databases? Why not use XML?

What is missing from XML:

- ✓ Consistency and integrity: XML representations are tree-like and often encourage redundant representations of data
- ✓ Efficient storage: XML is intended for data exchange, not data storage (arguable)
- ✓ Efficient query evaluation: current XML query technology does not scale, and type checking is difficult
- ✓ Concurrency control: updates, transactions and recovery are not well understood for XML – many people want to work on the data simultaneously
- ✓ Security: security policies should be enforced such that different users have permissions to access different subsets of the data
- ✓ . . .

The problems get worse when the data gets large

Example: a document for students and courses

What is wrong with the following?

```
<school>
  <student   id = "011">  <name> G. W. Bush   </name>
   <taking><course   cno = "Eng">  <title> Spelling</title>  </course>
           <course   cno = "CS2">  <title> XML    </title>  </course>
   </taking>
    <GPA>   1.5  </GPA>
  </student>
  <student   id = "012">  <name> D. Cheney   </name>
   <taking> <course cno = "CS2">   <title>politics</title> </course>
   </taking>
   <GPA>   1.4  </GPA>
</student>
  . . .
</school>
```


Problem 1. Data organization

Redundancy: It appears to be keeping our course records in student records and doing so *redundantly*

✓ efficiency: consider information about the courses that the students are taking, such as instructor, classroom, etc.

✓ consistency: the "CS2" record is different in Bush and Cheney's record

✓ loss of information: what if we drop Bush and Cheney from the document? We may end up losing all the information about CS2 and Eng

So maybe we want to separate our student records and course records, and link them. But how?

Problem 2. Efficiency

- Probably a student file would never contain more than a few thousand entries. But there are things we'd like to do quickly and efficiently – even with our simple files. Examples:
  - Give me all the students whose gpa is greater than 3.0.
  - Who are taking CS2 and Eng simultaneously?
- We would like to "program" these as quickly as possible.
- We would like these programs to be executed efficiently. What would happen if you were using a document of universities with hundreds of thousands of entries?

Problem 3. Concurrency control

Suppose several people are allowed to modify the document of student records. How do we stop two people changing the file at the same time and leaving it in a physical (or logical) mess?

Example. Suppose at the same time,

- Bush reads his gpa (1.5), adds 2.5 to it, and writes it back
- Cheney reads Bush's gpa, subtracts it by 1.5, and writes it back.

What would happen if Cheney reads Bush's gpa before Bush writes it, and writes the result back after Bush writes it? Bush's gpa now becomes 0 instead of 2.5!

If you don't care about your GPA, just imagine this might happen to your bank account.

Problem 4. Recovery and reliability

The system may crash while we are changing the document

Example. We are increasing everyone's gpa by a certain percentage in response to Blair's education reform (this may not happen), but in the middle of it, the system crashes. Then those who didn't get an increase would not be happy.

Problem 5. Consistency and security

- How to ensure
  - each student has a unique sid?
  - gpa is in the range from 0 to 4?
- How to prevent
  - a student from increasing his/her own gpa?
  - a student from checking other's gpa?

Operating systems are not sufficiently flexible to enforce security policies.

Advantages of a DBMS

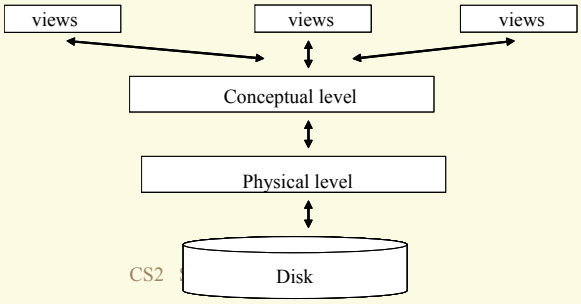
- ✓ Data independence: it provides an abstract view of data to insulate application code from details of data representations and storage.
- ✓ Uniform data administration and efficient data access: it utilizes a variety of techniques to store and retrieve data efficiently.
- ✓ Reduced development time: it supports many important functions that are common to many applications accessing data stored in the DBMS.

Advantages of a DBMS (cont'd)

- ✓ Concurrency control: it schedules concurrent accesses to the data such that the users can pretend they are using a single-user system.
- ✓ Recovery from crashes: it also protects users from the effects of system failure.
- ✓ Data integrity: it supports integrity constraints, that is, conditions that data must satisfy to ensure that the data is consistent and accurate.
- ✓ Security: it enforces access control on the data.

Three-level architecture

- The logical structure: what users see. The program or query language interface.
- The physical structure: how files are organized. What indexing mechanisms are used.
- Further the "logical" level is split into two: conceptual level and view level



Fix the problem of the example XML file (cont'd)

- Separate students from courses

```
<school>
  <student   id = "011">  <name> G. W. Bush </name> <gpa> 1.5  </gpa>
  </student>
  <student   id = "012">  <name> D. Cheney   </name> <gpa> 1.4  </gpa>
</student>
   . . .
  <course   cno = "Eng">  <title> Spelling</title>  </course>
  <course   cno = "CS2">  <title> XML     </title>  </course>
  . . .
</school>
```

Fix the problem of the example XML file

- Add enrollment records to link the two

```
<school>
  <student  id = “011”>  <name> G. W. Bush </name> <gpa> 1.5  </gpa>
  </student>
  <student  id = “012”>  <name> D. Cheney </name>  <gpa>  1.4  </gpa>
  </student>
   . . .
  <course   cno = “Eng”>  <title> Spelling</title>  </course>
  <course   cno = “CS2”>  <title> XML     </title>  </course>
  . . .
   <enroll     cno = “Eng”   sid = “011” />
   <enroll     cno = “CS2”    sid = “011”/>
   <enroll     cno = “CS2”   sid = “012”/>
  . . .
</school>
```


This ends up with three tables

- Conceptual level:

  Students(name: string, sid: string, email: string, gpa: real)

  Courses(title: string, cno: string)

  Enroll(sid: string, cno: string)
- This is an example of a relational database schema, which we will talk about soon.
  - Physical level:
    - Relations (a set of records) stored as unordered files
    - Index on the second column of students
  - View:

    Course-info(cno: string, enrollment: integer)

Data independence

- Logical data independence: protection from changes in logical structure of data.

  The ability to modify the conceptual level without causing the application programs (queries against views) to be rewritten.
- Physical data independence: protection from changes in physical structure of data.

  The ability to modify the physical level without causing changes at the conceptual level.

This is one of the most important benefits of using a DBMS.

After all, you don't worry about how numbers are stored when you use a computer-based calculator. This is the same principle.

Example SQL query

One should be able to use SQL to query the university database, for example,

```
SELECT   name
FROM     Students
WHERE    gpa > 3.0
```

without knowing, nor caring about how precisely data is stored

That's the traditional view, but …

Three-level architecture is not always achievable. When databases get big, users still have to worry about efficiency.

There are databases over which we have no control. For example, the Web is a giant, disorganized database.

There are also well-organized databases on the Web, for example,

http://www-db.stanford.edu/

which has a very nice organization, but for which the terminology does not quite apply.

Describing data in a DBMS – data models and database design

When we design a database we try to think "logically", but we need some kind of framework in which to design the database.

The problem is rather like designing a data structure in some programming language. You might use arrays, lists, etc. depending on what is available.

- A data model is a collection of concepts for describing data.

  Data model vs. type system in programming language.
- A schema is a description of a particular collection of data, using the given data model.

  Schemas vs. types in programming language.
- An instance of a schema (database) is the collection of data stored in the database at a particular moment in time.

  Instances vs. values of types.

The relational model – an introduction (1)

The **relational data model** is the most widely used model today.
Vendors: IBM, Microsoft, Oracle, Sybase, etc.
Recall the notions of sets and records.

- **Relation**: a table with rows and columns (or a *set* of tuples).
  - **Column**: fields, attributes
  - **Rows**: tuples, records

name	Sid	email	gpa
John	0001	John@nimbus.ocis	3.0
Joe	0002	Joe@nimbus.ocis	3.6
Mary	0003	Mary@nimbus.ocis	2.8
Grace	0004	Grace@nimbus.ocis	4.0

The relational model – an introduction (2)

- (Relation) Schema: every relation has a schema, which describes the columns (fields, attributes).

Example:

Students(name: string, sid: string, email: string, gpa: real)
Courses(title: string, cid: string, credits: integer)
Enroll(sid: string, cid: string, grade: string)

- Relational database: a collection of relations with distinct relation names.
- Database schema: a collection of relation schemas for the relations in the database.

Example: {Students, Courses, Enroll}

Why are relational databases so important?

- ✓ They are extremely simple to understand
- ✓ Query languages for them are well understood. There is an interesting and useful connection between relational query languages and first-order logic
- ✓ Query languages are optimizable. There is well-developed technology for this. Optimizing queries for hierarchical and XML data is much less well-understood.
- ✓ Updates and transaction processing are easier to understand and implement for relational databases.

Other data models

- Object-oriented data model.
  Object-oriented database systems: ObjectStore, O2, Ode, etc.
- Object-relational model.
  Object-relational database systems: UniSQL, Informix Universal Server, etc.
- Semantic data model, e.g., the ER model.
- Semistructured data models: OEM, XML trees (DOM), etc.
  Developed for Web databases and for data on the Web.
- Other data models:
- Network model
- Hierarchical model
- The functional data model
- …

Database languages

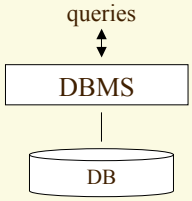
- **Data definition language** (**DDL**): defining schemas.
- **Data manipulation language** (**DML**): retrieving and manipulating data.
  - **Query**: a statement requesting the retrieval of information.

    Example: a SQL query

    SELECT name
    FROM Students
    WHERE gpa > 3.0
  - **Query language**: expressing queries. It is part of DML.
  - Relational query languages:
    - relational algebra and relational calculus
    - Commercial languages: SQL (Structured Query Language) and QBE (Query-By-Example).
  - **Updates**: insert, delete, modify.

Structure of a DBMS

- DDL and DML compilers, query processor.
- Query processor: the most important components of a DBMS:
  - query execution plan
  - query optimization
- File and access methods
- Buffer management
- Disk space management
- Concurrency control and recovery



Database folks -- where do you want to end up?

- **Database implementers:** build DBMS software.
- **End-users**: store and use data in a DBMS.

  Query languages, conceptual level design, ect. The basic skills have become a must for many jobs today.
- **Database application programmers**: develop packages that facilitate data access for a particular group of end-users.

  Query languages, conceptual and physical level design, system functions, etc.
- **Database administrators**: design and maintain databases. Conceptual and physical level design: security and authorization, recovery from failure, database tuning
- Others: the Web people (e.g., E-commerce, XML, workflows) .

What we shall learn

- Relational data model
- Relational query languages
  - Relational algebra
  - SQL

Summary – what you should remember!

- ✓ Database and DBMS.

Why DBMS? What is the typical structure of a DBMS?

- ✓ Data independence. Levels of abstraction (logical, physical).

What is logical independence? Physical independence? Why?

- ✓ Data models. Schemas. Instances (database).

How to represent information about the real world in a database?

- ✓ Database languages: DDL, DML, query languages.

Is DML a language just for database updates? Is query language part of DML?

- ✓ DBA, end-user, application programmers.

What are the responsibilities of a DBA?