Lecture 5: XML

Tuesday, January 16, 2001

Outline

- XML, DTDs (*Data on the Web*, 3.1)
- Semistructured data in XML (3.2)
- Exporting Relational Data in XML (8.3.1)

Facts About XML

- 132 books at Amazon
- 875,340 pages at www.altavista.com
- Every database vendor Z has www.Z.com/xml
- Many applications are just fancier Websites
- But, most importantly, XML enables data sharing on the Web – hence our interest

XML

- eXtensible Markup Language
- XML 1.0 – a recommendation from W3C, 1998
- Roots: SGML (a very nasty language).
- After the roots: a format for sharing *data*

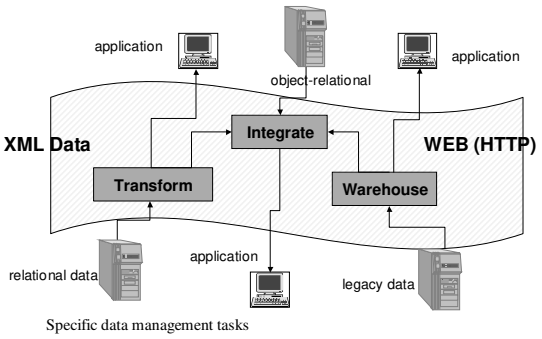
XML Applications

- Sharing data between different components of an application.
- Format for storing all data in Office 2000.
- Format for CISCO routers system tables.
- Format for EDI: electronic data exchange:
  - Transactions between banks
  - Producers and suppliers sharing product data (auctions)
  - Extranets: building relationships between companies
  - Scientists sharing data about experiments.

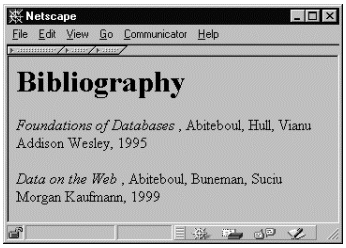
Why XML is of Interest to Us

- XML is just syntax for data
  - Note: we have no syntax for relational data
  - But XML is not relational: *semistructured*
- This is exciting because:
  - Can translate *any* legacy data to XML
  - Can ship XML over the Web (HTTP)
  - Can input XML into any application
  - Thus: data sharing and exchange on the Web

XML Data Sharing and Exchange



What is XML ?
From HTML to XML



HTML describes the presentation: easy for humans

HTML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
      Abiteboul, Hull, Vianu
      <br> Addison Wesley, 1995
<p> <i> Data on the Web </i>
      Abiteboul, Buneman, Suciu
      <br> Morgan Kaufmann, 1999
```

HTML is hard for applications

XML

```
<bibliography>
    <book>   <title> Foundations… </title>
             <author> Abiteboul </author>
             <author> Hull </author>
             <author> Vianu </author>
             <publisher> Addison Wesley </publisher>
             <year> 1995 </year>
    </book>
    …
</bibliography>
```

XML describes the content: easy for applications

XML Syntax

- Another example:

```
<db>
 <book>
    <title>Complete Guide to DB2</title>
    <author>Chamberlin</author>
 </book>
 <book>
    <title>Transaction Processing</title>
    <author>Bernstein</author>
    <author>Newcomer</author>
 </book>
 <publisher>
   <name>Morgan Kaufman</name>
   <state>CA</state>
 </publisher>
</db>
```

XML Terminology

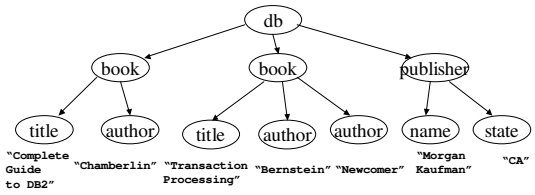
- tags: book, title, author, …
- start tag: <book>, end tag: </book>
- start tags must correspond to end tags, and conversely

XML Terminology

- an element: everything between tags
  - example element:
    <title>Complete Guide to DB2</title>
  - example element:
    <book> <title> Complete Guide to DB2 </title>
    <author>Chamberlin</author>
    </book>
- elements may be nested
- empty element: <red></red> abbreviated <red/>
- an XML document has a unique root element

well formed XML document: if it has matching tags

The XML Tree



Tags on nodes
Data values on leaves

More XML Syntax: Attributes

<book price = "55" currency = "USD">
  <title> Complete Guide to DB2 </title>
  <author> Chamberlin </author>
  <year> 1998 </year>
</book>

price, currency are called *attributes*

Replacing Attributes with Elements

<book>
  <title> Complete Guide to DB2 </title>
  <author> Chamberlin </author>
  <year> 1998 </year>
  <price> 55 </price>
  <currency> USD </currency>
</book>

attributes are alternative ways to represent data

"Types" (or "Schemas") for XML

- Document Type Definition – DTD
- Define a *grammar* for the XML document
  - we use it as substitute for types/schemas
- Will be replaced by XML-Schema

An Example DTD

```
<!DOCTYPE db [
  <!ELEMENT db   ((book|publisher)*)>
  <!ELEMENT book (title,author*,year?)>
  <!ELEMENT title      (#PCDATA)>
  <!ELEMENT author  (#PCDATA)>
  <!ELEMENT year  (#PCDATA)>
  <!ELEMENT publisher  (#PCDATA)>
]>
```

- PCDATA means *Parsed Character Data* (a mouthful for *string*)

DTDs as Grammars

```
db          ::=  (book|publisher)*
book        ::=  (title,author*,year?)
title       ::=  string
author      ::= string
year        ::= string
publisher   ::= string
```

- A DTD is a EBNF (Extended BNF) grammar
- An XML tree is precisely a derivation tree

XML Documents that have a DTD and conform to it are called valid

More on DTDs as Grammars

```
<!DOCTYPE paper [
  <!ELEMENT paper   (section*)>
  <!ELEMENT section ((title,section*) | text)>
  <!ELEMENT title     (#PCDATA)>
  <!ELEMENT text      (#PCDATA)>
]>
```

```
<paper> <section> <text> </text> </section>
        <section> <title> </title> <section> … </section>
                                   <section> … </section>
        </section>
</paper>
```

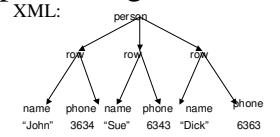
XML documents can be nested arbitrarily deep

XML for Representing Data

person

name	phone
John	3634
Sue	6343
Dick	6363

XML:




```
<person>
    <row> <name>John</name>
          <phone> 3634</phone></row>
    <row> <name>Sue</name>
          <phone> 6343</phone>
    <row> <name>Dick</name>
          <phone> 6363</phone></row>
</person>
```

DTDs as Schemas

- The DTD is:

```
<!DOCTYPE db [
  <!ELEMENT db   (person, department, product)>
  <!ELEMENT person   (row*)>
  <!ELEMENT row (name,phone)>
  <!ELEMENT name   (#PCDATA)>
  <!ELEMENT phone  (#PCDATA)>
]>
```

XML vs Other Data Models

- XML is self-describing
- Schema elements become part of the data
  - Reational schema: person(name,phone)
  - In XML <person>, <name>, <phone> are part of the data, and are repeated many times
- Consequence: XML is much more flexible, but also more inefficient
- XML = semistructured data

Semi-structured Data Explained

- Missing attributes:
  - <person> <name> John</name>
    <phone>1234</phone>
    </person>
  - <person><name>Joe</name></person> no phone !
- Repeated attributes
  - <person> <name> Mary</name>
    <phone>2345</phone>
    <phone>3456</phone>
    </person>

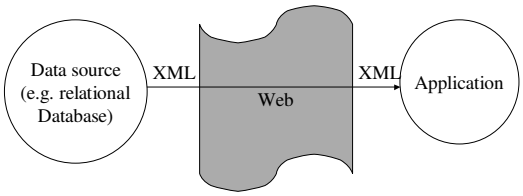
Semistructured Data Explained

- Attributes with different types in different objects
  - <person> <name> <first> John </first>
    <last> Smith </last>
    </name> complex name !
    <phone>1234</phone>
    </person>
- Nested collections (no 1NF)
- Heterogeneous collections:
  - <db> contains both <book>s and <publisher>s

XML Data v.s. E/R, Relational

- Q: is XML better or worse ?
- A: serves different purposes
  - E/R, Relational models:
    - For centralized processing, when we control the data
  - XML:
    - Data sharing between different systems
    - we do not have control over the entire data
- Do NOT use XML to *model* your data ! Use E/R, ODL, or relational instead.

Data Sharing with XML: Easy ☺



Projects

- All (except one) are related to XML
- All are potential research projects
  - Some of your colleagues already do research on these topics
- Readings have been added to the Website

Project 1: Indexing patterns

- An XML-QL pattern:

```
<bib> <book> <author> Abiteboul </author>
              <title> $x </title>
              <year> $y </year>
       </book>
</bib>
```

- Looking for titles, years of all books published by Abiteboul.
- Problem: given a large XML file, preprocess it in order to answer quickly *any* pattern
- Goal of the project: implement 2-3 simple methods, evaluate and compare them.
- Notice: Pradeep Shenoy is working on this

Project 2: Xpath containment

- Xpath allow us to express simple patterns, with a single variable. E.g.
  /bib/book[author=Abiteboul, year=1999]/title
- Some queries are "contained" in others. E.g. the above is contained in:
  //*[author=Abiteboul]//*[year=1999]/title
- Containment for the full Xpath language is probably complex
- Goal: define a "reasonable" subset of Xpath, solve the containment problem, implement it.
- Notice: Gerome Miklau is working on this

Project 3: Xpath query pruning

- We are given an Xpath query and a DTD or XML-Schema. E.g.:
  Xpath query: //editor
  DTD: [says that there are papers and books in the document, but only books may have an editor]
- Rewrite the Xpath query to the "more efficient": /bib/book/editor
- Goal: for a fragment of Xpath and of DTDs (or XML-Schema), find an algorithm for pruning queries in that fragment against schemas; implement it.

Project 4: Storing XML as relational data

- Given an XML document, therea re three ways to store it in relations (see papers)
- Goal of the project: evaluate the three alternatives on some large XML data instance (to be provided). Use SQL server, or some other DBMS

Project 5: Publishing relational data as XML

- Two research prototypes are published in the literature (see references)
- How do commercial products do it ?
- Goal: do a study of how commercial products approach XML publishing. Implement query rewriting for one of them (say SQL Server)

Project 6: Bulk Processing of Recursive Transformations

- Today's most popular XML language is XSLT: top-down, recursive processing
- How can we "translate" XSLT to SQL ? We can't.
- For a nice subset ("structural recursion") this is possible, using a sophisticated (read *inefficient*) technique: epsilon-edges.
- Goal: choose a subset of structural recursion which can be translated efficiently to SQL, and implement the translation.

Project 7: Processing Ordered Collections in SQL

- Goal: design an extension of SQL with ordered collections, translated it into SQL over relations with an "index" attribute.
- Notice: Yana Kadiyska is working on this.