Lecture 8: XML Data

Wednesday, October 11 2000

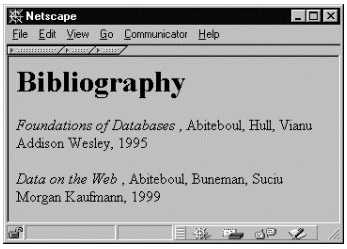
Outline

- 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 X has www.x.com/xml
- Many applications are just fancier Websites
- But, most importantly, XML enables data sharing on the Web – hence our interest

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

- eXtensible Markup Language
- Roots: comes from SGML (very nasty language).
- After the roots: a format for sharing *data*
- Emerging format for data exchange on the Web and between applications

XML Applications

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

XML Syntax

- Very simple:

```
<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

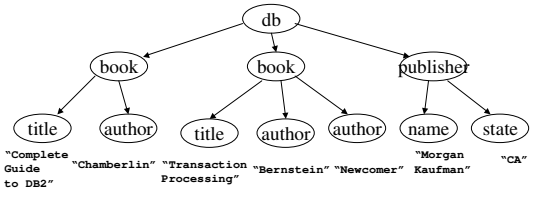
- tags: book, title, author, …
- start tag: <book>, end tag: </book>
- start tags must correspond to end tags, and conversely

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 (worse ☺) to represent data

"Types" (or "Schemas") for XML

- Document Type Definition – DTD
- Define a *grammar* for the XML document, but we use it as substitute for types/schemas
- Will be replaced by XML-Schema (will extend DTDs)

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>
```

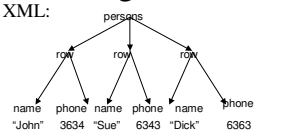
XML documents can be nested arbitrarily deep

XML for Representing Data

persons

name	phone
John	3634
Sue	6343
Dick	6363

XML:




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

XML vs Data Models

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

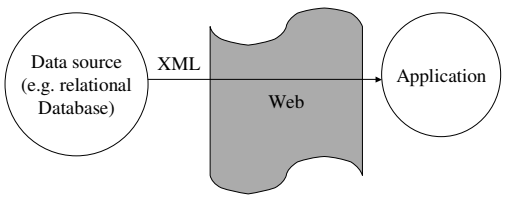
Semistructured Data Explained

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

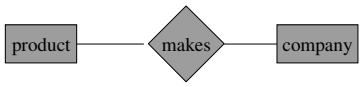
XML Data v.s. E/R, ODL, Relational

- Q: is XML better or worse ?
- A: serves different purposes
  - E/R, ODL, 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
    - E.g. on the Web
- Do NOT use XML to model your data ! Use E/R, ODL, or relational instead.

Data Sharing with XML: Easy ☺



Exporting Relational Data to XML



- Product(pid, name, weight)
- Company(cid, name, address)
- Makes(pid, cid, price)

Export data grouped by companies

```
<db><company>   <name> GizmoWorks </name>
                <address> Tacoma </address>
                <product> <name> gizmo </name>
                          <price> 19.99 </price>
                </product>
                <product> …</product>
        …
    </company>
    <company>   <name> Bang </name>
                <address> Kirkland </address>
                 <product> <name> gizmo </name> <price> 22.99 </price>
                 </product>
                 …
    </company>
    …
</db>
```

Redundant representation of products

The DTD

```
<!ELEMENT db (company*)>
<!ELEMENT company (name, address, product*)>
<!ELEMENT product (name,price)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

Export Data by Products

```
<db> <product> <name> Gizmo </name>
               <manufacturer>
                    <name> GizmoWorks </name>
                    <price> 19.99 </price>
                    <address> Tacoma </address>
               </manufacturer>
               <manufacturer>
                     <name> Bang </name>
                     <price> 22.99 </price>
                     <address> Kirkland </address>
               </manufacturer>
               …
     </product>
     <product> <name> OneClick </name> …
</db>
```

Redundant representation of companies

Which One Do We Choose ?

- The structure of the XML data is determined by agreement, with our partners, or dictated by committees
  - Many XML dialects (called *applications*)
- XML Data is often nested, irregular, etc
- No normal forms for XML ☺

Storing XML Data

- We got lots of XML data from the Web, how do we store it ?
- Ideally: convert to relational data, store in RDBMS
- Much harder than exporting relations to XML (why ?)
- DB Vendors currently work on tools for loading XML data into an RDBMS