XML Outline

- **XML, DTDs**
- **Semistructured data in XML**
- **Exporting Relational Data in XML**

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***

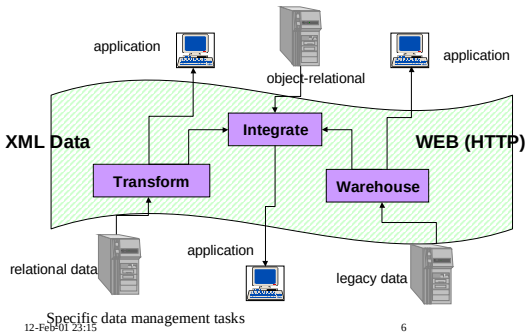
XML Applications

- **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.

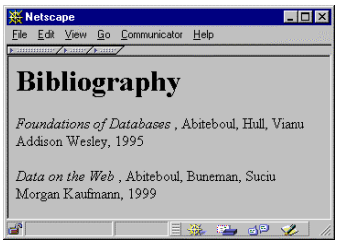
Why XML is of Interest to Us

- **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

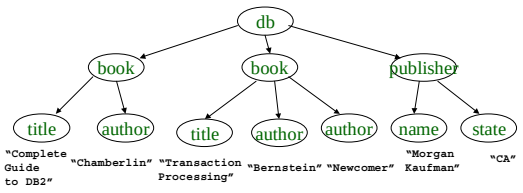
- **tags: book, title, author, …**
- **start tag: <book>, end tag: </book>**
- **start tags must correspond to end tags, and conversely**

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

"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>
```

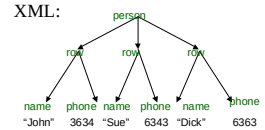
XML documents can be nested arbitrarily deep

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>

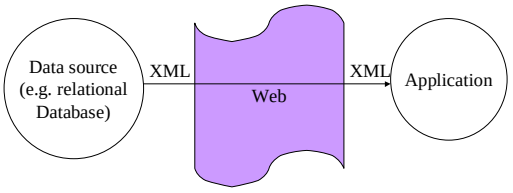
Semistructured Data Explained

- **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

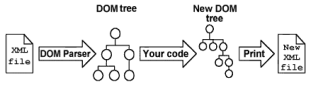
- **Q: is XML better or worse ?**
- **A: serves different purposes**
  - E/R, Relational models:
    - For centralized processing, when we control the data
  - XML:
    - Data sharing between different systems
    - we do not have control over the entire data
- **Do NOT use XML to *model* your data !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

Data Sharing with XML: Easy ☺



Document Object Model (DOM) in XML

- An XML structured document can be treated and manipulated as an object
- DOM parser transforms the document into a parse tree



- Program can walk the tree, performing arbitrary transformations
- DOM API then converts the new tree to another XML file
- New file can be printed, sent over the net or used as input to another program
- Applications can now exchange data without knowing formats — DTD contains everything necessary

XML to HTML

- **Fantasy: keep my content in XML (or in DB).**
  - Map to HTML on-the-fly based on browser.
- **Reality 1: dynamic data (e.g., stock quotes).**
  - There is no choice.
- **Reality 2: will static HTML go away?**
  - No, because this process is expensive to execute on the server.
  - Client-side rendering ought to help!

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: Storing XML as relational data

- **Given an XML document, there are 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 3: 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)**