Introduction to Database Systems
CSE 444
Lecture 10
XML
October 17, 2007

XML Outline
• XML (4.6, 4.7)
  – Syntax
  – Semistructured data
  – DTDs

Further Readings on XML
• Main source on XML, but hard to read:
  – http://www.w3.org/XML/
• Two tutorials out of myriads:

You don’t need to read this for the class

Additional Readings on XPath/XQuery
• Recommended reading on Xquery
  – http://www.w3.org/TR/xquery-use-cases/
• Other suggested readings:
  – http://www.w3.org/TR/xquery/
  – http://www.galaxquery.org/

Note: XML/XQuery is NOT covered in the textbook

XML
• A flexible syntax for data
• Used in:
  – Data exchange
  – Flexible databases: e.g. property lists
  – Configuration files: e.g. Web.Config
  – Document markup: e.g. XHTML
• Roots: SGML - a very nasty language
We will study only XML as data

XML for Data Exchange
• Relational data does not have a syntax
  – I can’t “give” you my relational database
  – Examples of syntaxes: CSV (comma-separated-values), ASN.1
• XML = syntax for data
  – But XML is not relational: semistructured
• Usage:
  – Export: Database → XML
  – Transport/transform XML
  – Import: XML → Databases or application
XML for Databases

- Relational databases have rigid schema
  - Schema evolution is costly
- XML is flexible: semistructured data
  - Store data in XML
- Warning: not normal form! Not even 1NF
  - Don’t try this at home

From HTML to XML

HTML describes the presentation

XML Syntax

XML describes the content

XML Terminology

- tags: book, title, author, …
- elements are nested
- empty element: <red/></red> abbrv. <red/>
- an XML document: single root element

More XML: Attributes

well formed XML document: if it has matching tags

<book price="55" currency="USD">
  <title>Foundations of Databases</title>
  <author>Abiteboul</author>
  <year>1995</year>
</book>
Attributes v.s. Elements

Attributes are alternative ways to represent data

XML v.s. HTML

- What are the differences between XML and HTML?

In class

More XML: Oids and References

- Oids and references in XML are just syntax by someone who didn’t take 444
- Don’t use them: use your own foreign keys instead.

More XML: CDATA Section

- Syntax: &lt;![CDATA[ ......any text here...]]&gt;
- Example:

```xml
<example>
 &lt;![CDATA[ some text here &lt;/notAtag&gt; &lt;]]></example>
```

More XML: Entity References

- Syntax: &entityname;
- Example:

```xml
&lt;element&gt; this is less than &lt; &lt;/element&gt;
```
- Some entities:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>'</td>
<td>’</td>
</tr>
<tr>
<td>&quot;</td>
<td>“</td>
</tr>
<tr>
<td>'</td>
<td>“</td>
</tr>
<tr>
<td>'</td>
<td>“</td>
</tr>
</tbody>
</table>
More XML: Processing Instructions

- Syntax: `<?target argument?>`
- Example:

```
<product><name> Alarm Clock </name>
  <?ringBell 20?>
</product>
```

- What do they mean?

More XML: Comments

- Syntax `<!-- ... Comment text... -->`
- Yes, they are part of the data model!!!
Mapping Relational Data to XML Data

The canonical mapping:

<table>
<thead>
<tr>
<th>Persons</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>3634</td>
<td></td>
</tr>
<tr>
<td>Sue</td>
<td>6343</td>
<td></td>
</tr>
<tr>
<td>Dick</td>
<td>6363</td>
<td></td>
</tr>
</tbody>
</table>

XML:

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

Name Phone
John 3634
Sue 6343
Dick 6363

Mapping Relational Data to XML Data

Application specific mapping

<table>
<thead>
<tr>
<th>Persons</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>3634</td>
<td></td>
</tr>
<tr>
<td>Sue</td>
<td>6343</td>
<td></td>
</tr>
</tbody>
</table>

Orders

<table>
<thead>
<tr>
<th>PersonName</th>
<th>Date</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>2002</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Sue</td>
<td>2004</td>
<td>Gadget</td>
</tr>
</tbody>
</table>

XML is Semi-structured Data

- Missing attributes:

```
<person>
  <name>John</name>
  <phone>1234</phone>
</person>
```

- Could represent in a table with nulls

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1234</td>
</tr>
<tr>
<td>Joe</td>
<td>-</td>
</tr>
</tbody>
</table>

XML is Semi-structured Data

- Repeated attributes

```
<person>
  <name>Mary</name>
  <phone>3456</phone>
  <phone>2345</phone>
</person>
```

- Impossible in tables:

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>2345</td>
</tr>
</tbody>
</table>

XML is Semi-structured Data

- Attributes with different types in different objects

```
<person>
  <first>John</first>
  <last>Smith</last>
  <phone>1234</phone>
</person>
```

- Nested collections (no 1NF)
- Heterogeneous collections:
  - `<db>` contains both `<book>`s and `<publisher>`s

Document Type Definitions

DTD

- part of the original XML specification
- an XML document may have a DTD
- XML document:
  - **Well-formed** = if tags are correctly closed
  - **Valid** = if it has a DTD and conforms to it
  - validation is useful in data exchange

XML is Semi-structured Data

- Attributes with different types in different objects

```
<person>
  <first>John</first>
  <last>Smith</last>
  <phone>1234</phone>
</person>
```

- Nested collections (no 1NF)
- Heterogeneous collections:
  - `<db>` contains both `<book>`s and `<publisher>`s
**DTD**

Goals:
- Define what tags and attributes are allowed
- Define how they are nested
- Define how they are ordered

Superseded by XML Schema
- Very complex: DTDs still used widely

**Very Simple DTD**

Example of valid XML document:

```xml
<company>
  <person>  
    <ssn> 123456789 </ssn>
    <name> John </name>
    <office> B432 </office>
    <phone> 1234 </phone>
  </person>
  <person>  
    <ssn> 987654321 </ssn>
    <name> Jim </name>
    <office> B123 </office>
  </person>
  <product> ... </product>
...</company>
```

**DTD: The Content Model**

- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)*

**DTD: Regular Expressions**

- **sequence**
- **optional**
- **Kleene star**
- **alternation**