Introduction to Database Systems
CSE 444

Lecture 10
XML

April 21, 2008

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

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
HTML

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

XML Syntax

```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

XML Terminology

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

More XML: Attributes

```xml
<book price = “55” currency = “USD”>
  <title> Foundations of Databases </title>
  <author> Abiteboul </author>
  ...
  <year> 1995 </year>
</book>
```

well formed XML document: if it has matching tags
Attributes v.s. Elements

- Attributes are alternative ways to represent data.

Comparison

<table>
<thead>
<tr>
<th>Elements</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered</td>
<td>Unordered</td>
</tr>
<tr>
<td>May be repeated</td>
<td>Must be unique</td>
</tr>
<tr>
<td>May be nested</td>
<td>Must be atomic</td>
</tr>
</tbody>
</table>

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.

- Are just keys/foreign keys design by someone who didn’t take 444

- Don’t use them: use your own foreign keys instead.
More XML: CDATA Section

• Syntax: <![CDATA[ .....any text here...]]>

• Example:

```xml
<example>
  <![CDATA[ some text here </notAtag> <> ]]>
</example>
```

More XML: Entity References

• Syntax: &entityname;

• Example:

```xml
<element> this is less than &lt; </element>
```

• Some entities:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Character</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>&amp;38</td>
<td>Unicode char</td>
</tr>
</tbody>
</table>

More XML: Processing Instructions

• Syntax: <?target argument?>

• Example:

```xml
<product> <name> Alarm Clock </name> 
  <?ringBell 20?>
  <price> 19.99 </price>
</product>
```

• What do they mean?

More XML: Comments

• Syntax <!-- .... Comment text... -->

• Yes, they are part of the data model !!!
XML Namespaces

- name ::= [prefix:]localpart

    `<book xmlns:isbn="www.isbn-org.org/def">`
    `<title> ... </title>`
    `<number> 15 </number>`
    `<isbn:number> .... </isbn:number>`
    `</book>`

Means nothing as URL; just a unique name

XML Namespaces

- syntactic: `<number>`, `<isbn:number>`
- semantic: provide URL for schema

    `<tag xmlns:mystyle = "http://…">`
    `…`
    `<mystyle:title>… </mystyle:title>`
    `<mystyle:number> …`
    `</tag>`

XML Semantics: a Tree!

    `<data>`
      `<person id="o555">`
        `<name> Mary </name>`
        `<address>`
          `<street> Maple </street>`
          `<no> 345 </no>`
          `<city> Seattle </city>`
        `</address>`
      `</person>`
      `<person>`
        `<name> John </name>`
        `<address> Thailand </address>`
        `<phone>23456</phone>`
      `</person>`
    `</data>`

Order matters !!!

XML Data

- XML is self-describing
- Schema elements become part of the data
  - Relational 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
Mapping Relational Data to XML Data

The canonical mapping:

```
<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>
```

<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 is Semi-structured Data

- Missing attributes:

```
<person> <name>John</name> <phone>1234</phone> </person>
<person> <name>Joe</name> </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>2345</phone> <phone>3456</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>
<tr>
<td></td>
<td>3456</td>
</tr>
</tbody>
</table>
XML is Semi-structured Data

- Attributes with different types in different objects
  
  ```xml
  <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

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

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

```xml
<!DOCTYPE company [ 
  <!ELEMENT company ((person|product)*)> 
  <!ELEMENT person (ssn, name, office, phone?)> 
  <!ELEMENT ssn (#PCDATA)> 
  <!ELEMENT name (#PCDATA)> 
  <!ELEMENT office (#PCDATA)> 
  <!ELEMENT phone (#PCDATA)> 
  <!ELEMENT product (pid, name, description?)> 
  <!ELEMENT pid (#PCDATA)> 
  <!ELEMENT description (#PCDATA)> ]>
```
Very Simple DTD

Example of valid XML document:

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

DTD: The Content Model

```
<!ELEMENT tag (CONTENT)>
```

- 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: `<ELEMENT name (firstName, lastName)>`
- Optional: `<ELEMENT name (firstName?, lastName)>`
- Kleene star: `<ELEMENT person (name, phone*)>`
- Alternation: `<ELEMENT person (name, (phone|email))>`