

Lecture 10

XML

Monday, January 30, 2006

1

XML Outline

- XML (4.6, 4.7)
 - Syntax
 - Semistructured data
 - DTDs

2

Additional Readings on XML

Main source: www.w3.org (but hard to read)

- <http://www.w3.org/XML/>

Strongly recommend readings:

- <http://www.w3.org/XML/1999/XML-in-10-points>
- www.zvon.org/xxl/XMLTutorial/General/book_en.html

For XPath and XQuery (next lectures):

- <http://www.galaxquery.org/>

3

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

4

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

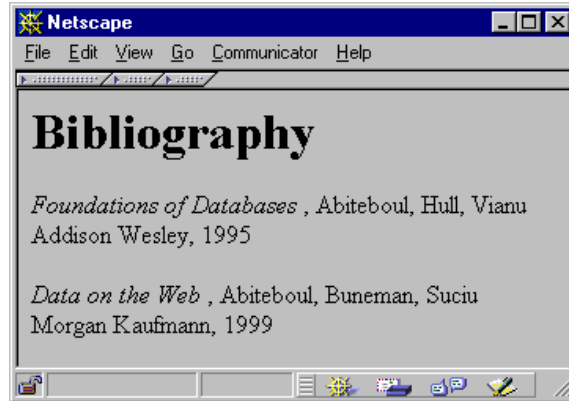
5

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

6

From HTML to XML



HTML describes the presentation

7

HTML

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

8

XML Syntax

```
<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

9

XML Terminology

- tags: `book`, `title`, `author`, ...
- start tag: `<book>`, end tag: `</book>`
- elements: `<book>...</book>`, `<author>...</author>`
- elements are nested
- empty element: `<red></red>` abbrev. `<red/>`
- an XML document: single *root element*

well formed XML document: if it has matching tags

10

More XML: Attributes

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

11

Attributes v.s. Elements

```
<book price = "55" currency = "USD">  
  <title> Foundations of DBs </title>  
  <author> Abiteboul </author>  
  ...  
  <year> 1995 </year>  
</book>
```

```
<book>  
  <title> Foundations of DBs </title>  
  <author> Abiteboul </author>  
  ...  
  <year> 1995 </year>  
  <price> 55 </price>  
  <currency> USD </currency>  
</book>
```

attributes are alternative ways to represent data

12

Comparison

Elements	Attributes
Ordered	Unordered
May be repeated	Must be unique
May be nested	Must be atomic

13

XML v.s. HTML

- What are the differences between XML and HTML ?

In class

14

More XML: Oids and References

```
<person id="o555">
  <name> Jane </name>
</person>

<person id="o456">
  <name> Mary </name>
  <mother idref="o555"/>
</person>
```

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

Don't use them: use your own
foreign keys instead.

oids and references in XML are just syntax

15

More XML: CDATA Section

- Syntax: `<![CDATA[.....any text here...]]>`
- Example:

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

16

More XML: Entity References

- Syntax: `&entityname;`
- Example:
`<element> this is less than < </element>`
- Some entities:

<code>&lt;</code>	<code><</code>
<code>&gt;</code>	<code>></code>
<code>&amp;</code>	<code>&</code>
<code>&apos;</code>	<code>'</code>
<code>&quot;</code>	<code>“</code>
<code>&#38;</code>	Unicode char

17

More XML: Processing Instructions

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

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

- What do they mean ?

18

More XML: Comments

- Syntax `<!-- Comment text... -->`
- Yes, they are part of the data model !!!

19

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

```

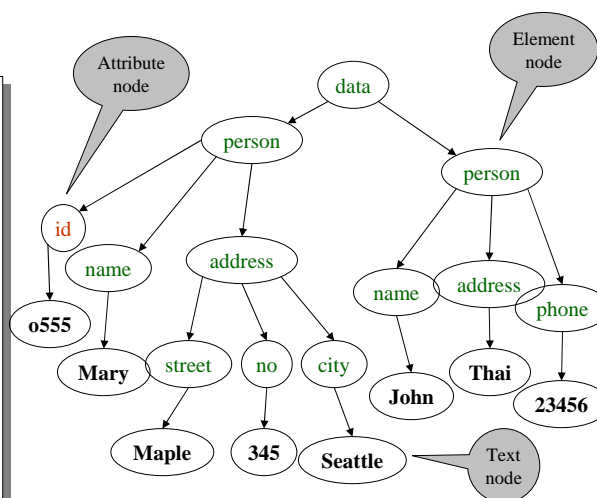
<tag xmlns:mystyle = "http://...">
  ...
  <mystyle:title>... </mystyle:title>
  <mystyle:number> ...
</tag>
  
```

21

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 !!!

22

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

23

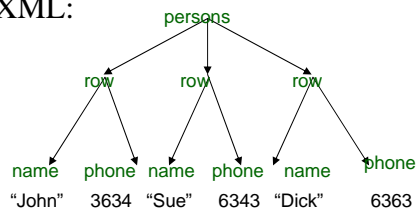
Mapping Relational Data to XML Data

The canonical mapping:

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>
```

24

Mapping Relational Data to XML Data

Application specific mapping

Persons

Name	Phone
John	3634
Sue	6343

Orders

PersonName	Date	Product
John	2002	Gizmo
John	2004	Gadget
Sue	2002	Gadget

XML

```

<persons>
  <person>
    <name> John </name>
    <phone> 3634 </phone>
    <order> <date> 2002 </date>
      <product> Gizmo </product>
    </order>
    <order> <date> 2004 </date>
      <product> Gadget </product>
    </order>
  </person>
  <person>
    <name> Sue </name>
    <phone> 6343 </phone>
    <order> <date> 2004 </date>
      <product> Gadget </product>
    </order>
  </person>
</persons>
  
```

XML is Semi-structured Data

- Missing attributes:

```

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

<person> <name>Joe</name>
</person>
  
```

no phone !

- Could represent in a table with nulls

name	phone
John	1234
Joe	-

26

XML is Semi-structured Data

- Repeated attributes

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

Two phones !

- Impossible in tables:

name	phone		???
Mary	2345	3456	

27

XML is Semi-structured Data

- Attributes with different types in different objects

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

Structured name !

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

28

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

29

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

30

Very Simple DTD

```
<!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)>  
>
```

31

Very Simple DTD

Example of valid XML document:

```
<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>
```

32

DTD: The Content Model

`<!ELEMENT tag (CONTENT)>`

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

DTD

XML

sequence

`<!ELEMENT name (firstName, lastName)>`



`<name>
<firstName> </firstName>
<lastName> </lastName>
</name>`

optional

`<!ELEMENT name (firstName?, lastName)>`

Kleene star

`<!ELEMENT person (name, phone*)>`

`<person>
<name> </name>
<phone> </phone>
<phone> </phone>
<phone> </phone>
.....
</person>`

alternation

`<!ELEMENT person (name, (phone|email))>`