

Introduction to Data Management CSE 344

Lecture 12: XML and XPath

CSE 344 - Fall 2011

1

XML Outline

- What is XML?
- Syntax
- Semistructured data
- DTDs
- XPath

CSE 344 - Fall 2011

2

What is XML?

- Stands for eXtensible Markup Language
 1. Advanced, self-describing file format
 2. Based on a flexible, semi-structured data model
- Applications:
 - Data exchange
 - Storing data without a rigid schema: advertisements
 - Configuration files: e.g. Web.Config
 - Document markup: e.g. XHTML

We will study only XML as data

3

Some XML Benefits

- Simple human-readable format
- Machine readable format
- Software- and hardware- independent way of storing data
- Flexible structure (we will see today)

CSE 344 - Fall 2011

4

Data Exchange

- Relational data does not have a syntax
 - I can't "give" you my relational database or parts of it
 - Need some file format:
 - CSV (comma-separated-values), ASN.1
- XML
 - Is a more advanced file format
 - Also has its own data model: *semistructured*
- Main idea: applications exchange information in XML

CSE 344 - Fall 2011

5

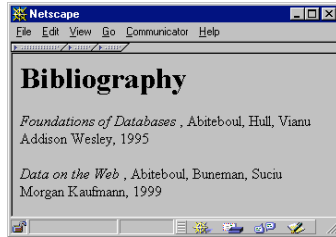
XML vs Relational

- Relational data model
 - Rigid flat structure (tables)
 - Schema must be fixed in advanced
 - Binary representation: good for performance, bad for exchange
 - Query language based on Relational Calculus
- Semistructured data model / XML
 - Flexible, nested structure (trees)
 - Does not require predefined schema ("self describing")
 - Text representation: good for exchange, bad for performance
 - Query language borrows from automata theory

CSE 344 - Fall 2011

6

From HTML to XML



HTML describes the presentation

CSE 344 - Fall 2011

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>
  Abiteboul, Buneman, Suci
  <br> Morgan Kaufmann, 1999
```

HTML describes the presentation

CSE 344 - Fall 2011

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

- Has matching tags
- A short header
- And a root element

10

Well-Formed XML

```
<? xml version="1.0" encoding="utf-8" standalone="yes" ?>
<SomeTag>
  ...
</SomeTag>
```

CSE 344 - Fall 2011

11

More XML: Attributes

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

CSE 344 - Fall 2011

12

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>
```

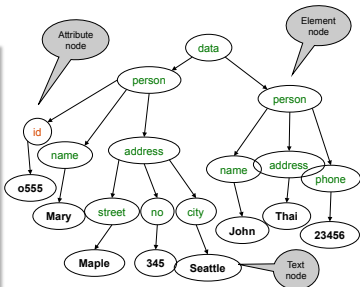
Attributes are alternative ways to represent data

Comparison

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

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: person(name,phone)
 - In XML <person>, <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:

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>
    <row> <name>Dick</name>
    <phone> 6363</phone></row>
  </person>
```

Mapping Relational Data to XML Data

Application specific mapping

Person	
Name	Phone
John	3634
Sue	6343

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

```
XML
  <people>
    <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>
</people>
```

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	-

CSE 344 - Fall 2011

19

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	

CSE 344 - Fall 2011

20

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
- Heterogeneous collections:
 - <db> contains both <book>s and <publisher>s

CSE 344 - Fall 2011

21

Schema

CSE 344 - Fall 2011

22

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
- Use <http://validator.w3.org/check> to validate

CSE 344 - Fall 2011

23

DTD

Goals:

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

Superseded by XML Schema (Book Sec. 11.4)

- Very complex: DTDs still used widely

CSE 344 - Fall 2011

24

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)>
]>
    
```

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>
    
```

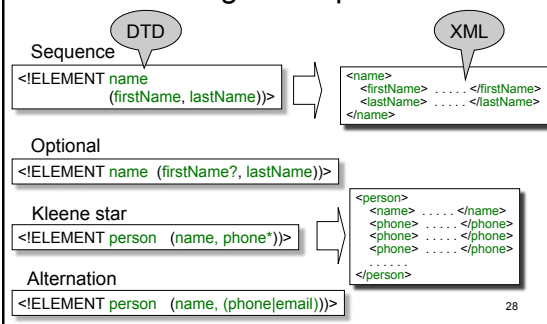
DTD: The Content Model

```
<!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: Attributes

```

From "sample-xml-with-dtd.xml"
<!DOCTYPE bib [
  <!ELEMENT bib (book*) >
  <!ELEMENT book (title, (author+ | editor+ ), publisher?, price) >
  <!ATTLIST book year CDATA #REQUIRED >
  ...
]>

<bib>
  <book year="1994">
  ...
    
```

DTD: Text

- Two options:
- #PCDATA ("Parsed Character Data") = the text inside elements
 - CDATA ("Character Data") = the text inside attributes
 - There is no #CDATA and no PCDATA

Cross-references

- An attribute is called an ID attribute if it is unique
- An attribute is called IDREF if it references an ID
- The DTD defines which attribute(s) are ID/IDREFs

From "mondial.dtd" (HW4)

```
<!ATTLIST country car_code ID #IMPLIED
  area CDATA #IMPLIED
  capital IDREF #IMPLIED
  memberships IDREFS #IMPLIED>
```

IMPLIED means that it does not have to be included

Querying

Querying XML Data

- XPath = simple navigation through the tree
- XQuery = the SQL of XML
- XSLT = recursive traversal
 - will not discuss in class

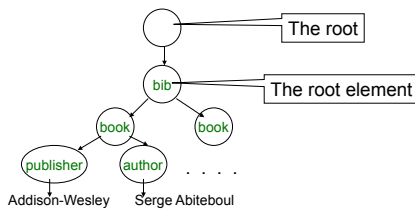
Sample Data for Queries

```
<bib>
  <book> <publisher> Addison-Wesley </publisher>
    <author> Serge Abiteboul </author>
    <author> <first-name> Rick </first-name>
      <last-name> Hull </last-name>
    </author>
    <author> Victor Vianu </author>
    <title> Foundations of Databases </title>
    <year> 1995 </year>
  </book>
  <book price="55">
    <publisher> Freeman </publisher>
    <author> Jeffrey D. Ullman </author>
    <title> Principles of Database and Knowledge Base Systems </title>
    <year> 1998 </year>
  </book>
</bib>
```

Data Model for XPath

XPath returns a sequence of items. An item is either:

- A value of primitive type, or
- A node (doc, element, or attribute)



XPath: Simple Expressions

`/bib/book/year`

Result: `<year> 1995 </year>`
`<year> 1998 </year>`

`/bib/paper/year`

Result: empty (there were no papers)

`/bib`

What's the difference ?

`/`

XPath: Restricted Kleene Closure

`//author`

Result: `<author> Serge Abiteboul </author>`
`<author> <first-name> Rick </first-name>`
`<author> <last-name> Hull </last-name>`
`</author>`
`<author> Victor Vianu </author>`
`<author> Jeffrey D. Ullman </author>`

`/bib//first-name`

Result: `<first-name> Rick </first-name>`

CSE 344 - Fall 2011

37

XPath: Attribute Nodes

`/bib/book/@price`

Result: "55"

`@price` means that price has to be an attribute

CSE 344 - Fall 2011

38

XPath: Wildcard

`//author/*`

Result: `<first-name> Rick </first-name>`
`<last-name> Hull </last-name>`

* Matches any element

@* Matches any attribute

CSE 344 - Fall 2011

39

XPath: Text Nodes

`/bib/book/author/text()`

Result: Serge Abiteboul
 Victor Vianu
 Jeffrey D. Ullman

Rick Hull doesn't appear because he has `first-name`, `last-name`

Functions in XPath:

- `text()` = matches the text value
- `node()` = matches any node (= * or @* or `text()`)
- `name()` = returns the name of the current tag

CSE 344 - Fall 2011

40

XPath: Predicates

`/bib/book/author[first-name]`

Result: `<author> <first-name> Rick </first-name>`
`<author> <last-name> Hull </last-name>`
`</author>`

CSE 344 - Fall 2011

41

XPath: More Predicates

`/bib/book/author[first-name][address[//zip][city]]/last-name`

Result: `<last-name> ... </last-name>`
`<last-name> ... </last-name>`

How do we read this ?

First remove all qualifiers (predicates):

`/bib/book/author/last-name`

Then add them one by one:

`/bib/book/author[first-name][address]/last-name` etc

CSE 344 - Fall 2011

42

XPath: More Predicates

CSE 344 - Fall 2011 43

XPath: Position Predicates

The 2nd book
 The last book
 The 2nd of all books in 1998
 2nd book IF it is in 1998

CSE 344 - Fall 2011 44

XPath: More Axes

. means *current node*
 Same as
 Same as

CSE 344 - Fall 2011 45

XPath: More Axes

.. means *parent node*
 Same as

 Same as
Hint: don't use ..

CSE 344 - Fall 2011 46

XPath: Summary

bib * / /bib bib/paper bib//paper //paper paper book @price bib/book/@price bib/book[@price<"55"/author/last-name bib/book[@price<"55" or @price>"99"/author/last-name	matches a bib element matches any element matches the root element matches a bib element under root matches a paper in bib matches a paper in bib , at any depth matches a paper at any depth matches a paper or a book matches a price attribute matches price attribute in book , in bib matches... matches...
---	--

CSE 344 - Fall 2011 47