Introduction to Database Systems
CSE 414

Lecture 14-15: XML

Announcements

- Homework 4 solution will be posted tomorrow
- Midterm: Monday in class
  - Open books, no notes beyond one *hand-written* sheet of paper (2 sides ok), closed computers, etc.
  - Material: everything through basics of query execution (although light on physical operators – we'll do more with it on the next hw assignment)
  - Last-minute review Q&A Sunday, 2 pm, Loew 101

The Semistructured Data Model

- So far we have studied the *relational data model*
  - Data is stored in tables(=relations)
  - Queries are expressions in SQL (or relational algebra, or related languages…)
- Today: Semi-structured data model
  - XML is a very popular syntax for this data model
  - Best suited for data exchange

XML Outline

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

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

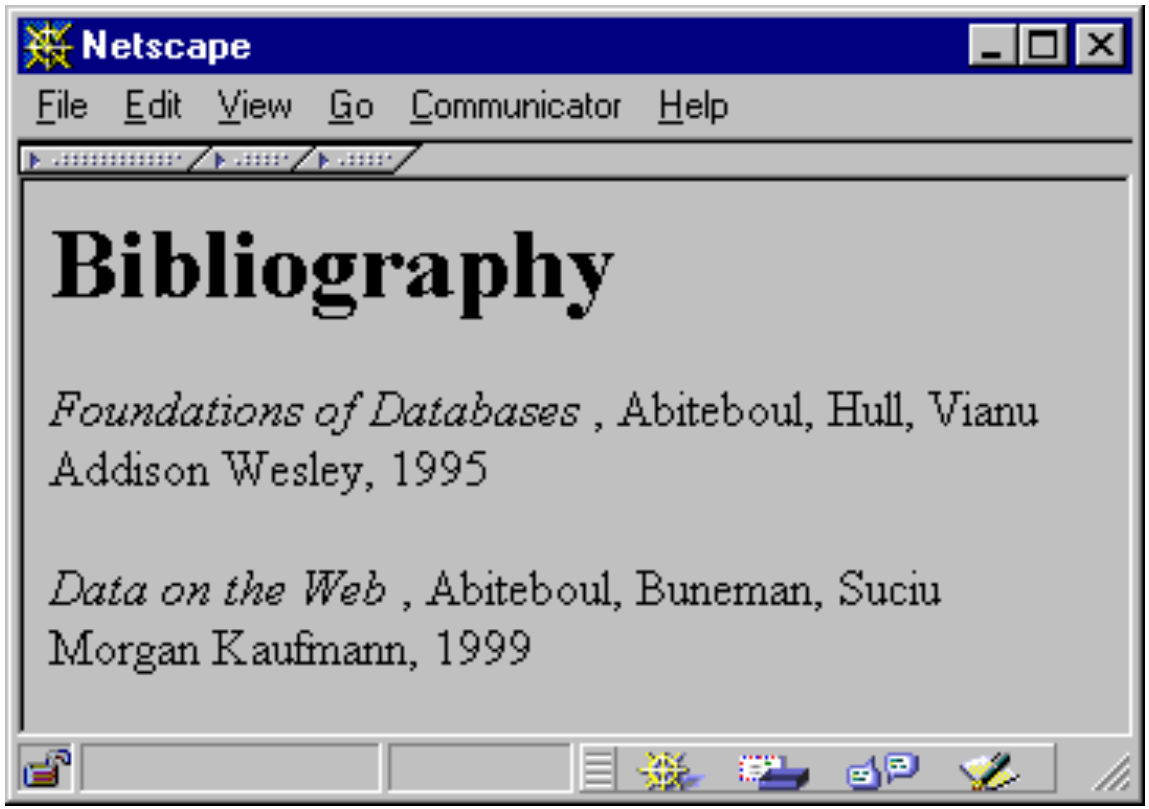
We will study only XML as data

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

From HTML to XML



HTML describes the presentation

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 describes the presentation

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

XML Terminology

- Tags: book, title, author, …
- Start tag: <book>, end tag: </book>
- Elements: <book>…</book>,<author>…</author>
- Elements are nested
- Empty element: <red></red> abbrv. <red/>
- An XML document: single *root element*

Well formed XML document
- Has matching tags
- A short header
- And a root element

Well-Formed XML

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

More XML: Attributes

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

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>
```

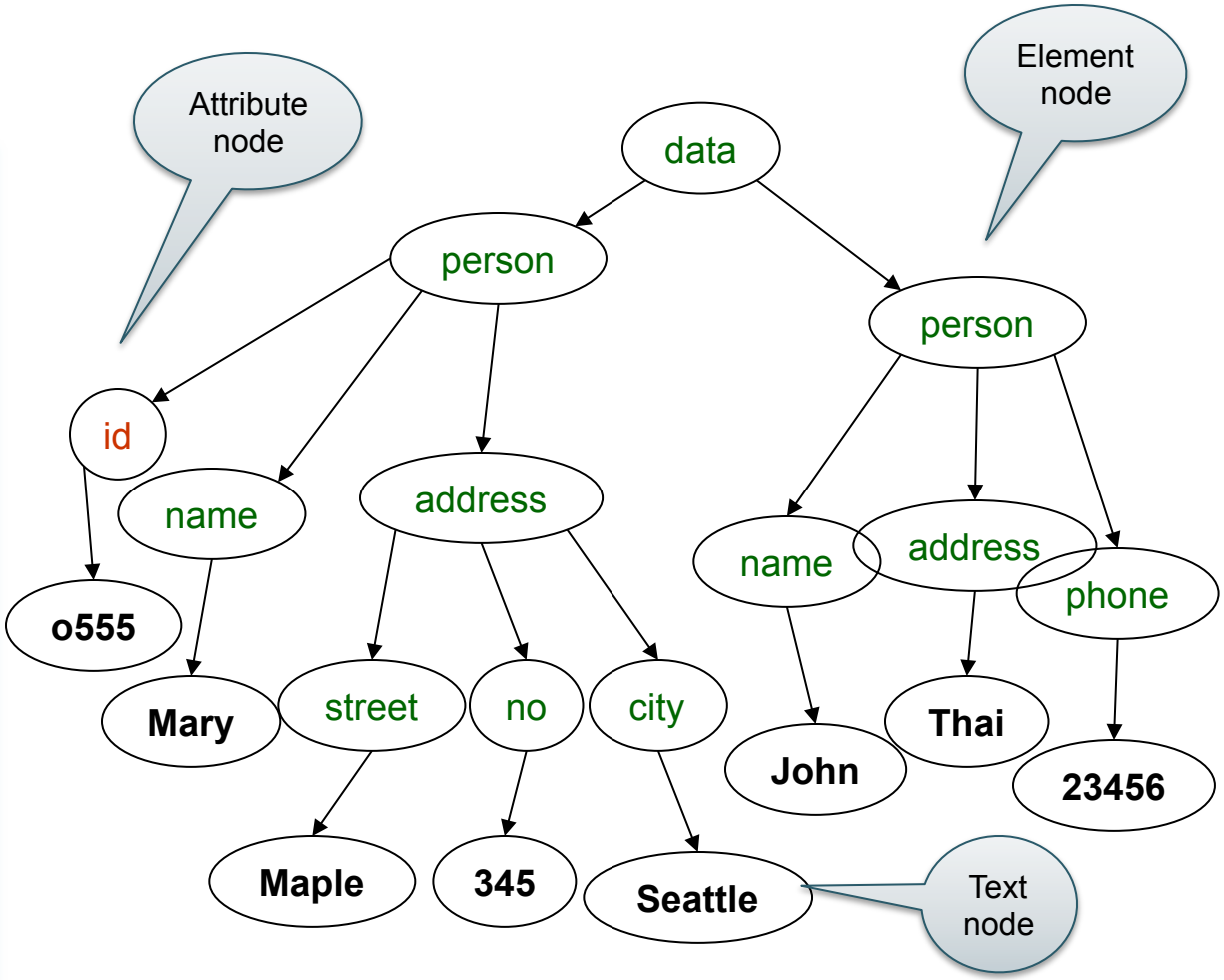
Attributes are alternative ways to represent data

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 !!!

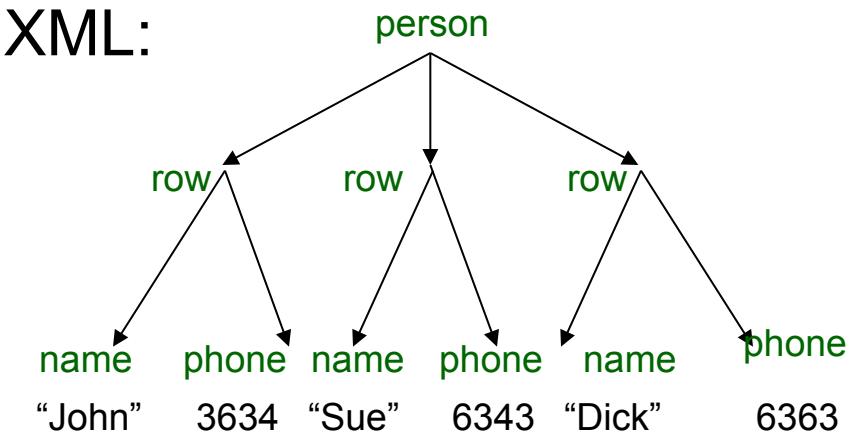
XML Data

- 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:

XML:



Person

Name	Phone
John	3634
Sue	6343
Dick	6363

```
<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=Semi-structured Data (1/3)

- 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	-

XML=Semi-structured Data (2/3)

- Repeated attributes

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

Two phones !

- Impossible in tables:

name	phone	
Mary	2345	3456

???

XML=Semi-structured Data (3/3)

- 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

Schema

Document Type Definitions (DTD)

- 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

Superseded by XML Schema (Book Sec. 11.4)

- Very complex: DTDs still used widely

Example 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)>
]>
```

Example 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

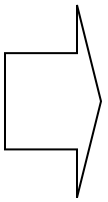
- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)*

DTD: Complex Content

DTD

Sequence

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



XML

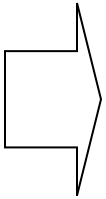
```
<name>
   <firstName>  . . . . . </firstName>
   <lastName>  . . . . . </lastName>
</name>
```

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)))>
```

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

Up next: XML queries

- XPath
- XQuery
- And a mention of JSON