Introduction to Data Management
CSE 344

Lecture 13: XML and XPath
Announcements

• Current assignments:
  – Web quiz 4 due next Tuesday, 11 pm
  – Homework 4 due Thursday, 11 pm

• Yuqing will hold extra office hours this Friday 1:30-2:30 in the lab

• My office hours today: 4:30-5:15

• Midterm: next Monday in class
  – Review Sunday, 3-5 pm in EEB 037
  – Includes everything up to and including today’s lecture

• Today’s lecture: secs. 11.1-11.3, 12.1
Here are **unsafe** datalog rules. What’s “unsafe” about them?

\[
U1(x,y) :\text{-} \text{Movie}(x,z,1994),\ y>1910
\]

\[
U2(x) :\text{-} \text{Movie}(x,z,1994),\ \text{not Casts}(u,x)
\]

A datalog rule is **safe** if every variable appears in some positive relational atom.
Domain Independent Relational Calculus

• As in datalog, one can write “unsafe” RC queries; they are also called *domain dependent*

\[
\begin{align*}
A(x) &= \text{not Likes("Fred", x)} \\
A(x,y) &= \text{Likes("Fred", x) OR Serves("Bar", y)} \\
A(x) &= \forall y. \text{Serves}(x,y)
\end{align*}
\]

• Lesson: make sure your RC queries are domain independent (only depends on database)
More Examples

Find drinkers that frequent some bar that serves some beer they like.

\[ Q(x) = \exists y. \exists z. \text{Frequents}(x, y) \land \text{Serves}(y, z) \land \text{Likes}(x, z) \]

Find drinkers that frequent only bars that serves some beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow (\exists z. \text{Serves}(y, z) \land \text{Likes}(x, z)) \]

Find drinkers that frequent some bar that serves only beers they like.

\[ Q(x) = \exists y. \text{Frequents}(x, y) \land \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z)) \]

Find drinkers that frequent only bars that serves only beer they like.

\[ Q(x) = \forall y. \text{Frequents}(x, y) \Rightarrow \forall z. (\text{Serves}(y, z) \Rightarrow \text{Likes}(x, z)) \]
XML and semistructured data model
The Semistructured Data Model

• So far we have studied the *relational data model*
  – Data is stored in tables (=relations)
  – Queries are expressions in the relational calculus (or relational algebra, or datalog, or SQL…)

• Today: Semi-structured data model
  – XML is one popular syntax for this data model
  – Other popular standards: JSON, PROTOBUF
XML Outline

• What is XML?
• Syntax
• Semistructured data
• DTDs
• XPath
• XQuery (next time)
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!
  – Also: storing data without a rigid schema; configuration files; XHTML
• Other popular data exchange formats:
  – JSON, PROTOBUF

We will study only XML, and 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

- *Foundations of Databases*, Abiteboul, Hull, Vianu
  - Addison Wesley, 1995

- *Data on the Web*, Abiteboul, Buneman, Suciu
  - Morgan Kaufmann, 1999
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> 
   Abiteboul, Buneman, Suciu 
   <br> Morgan Kaufmann, 1999 
</p>

HTML describes the presentation
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, …
• 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

```xml
<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
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 Semantics: a Tree!

```
<data>
  <person id="0555">
    <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:

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

XML:

```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

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

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

XML

```xml
<people>
  <person>
    <name>John</name>
    <phone>3634</phone>
    <order>
      <date>2002</date>
      <product>Gizmo</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>
```

- 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=Semi-structured Data (2/3)

• Repeated attributes

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

• Impossible in one table:

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>2345</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>3456</td>
</tr>
</tbody>
</table>

Two phones!
XML = Semi-structured Data (3/3)

- Attributes with different types in different objects

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

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