Semistructured Content

1

On our first day …

- Structured data : database system
  - tagged, typed
  - well-defined semantic interpretation
- Semi-structured data: tagged
  - XML (HTML?)
  - some help with semantic interpretation
- Unstructured: information retrieval
  - Text
  - Graphics: 2D, 3D
  - Music
  - Video
  - any help with semantic interpretation?

2

XML
eXtensible Markup Language

- General-purpose description of content of a document
- Includes namespaces → linking across the Web
- Designed by working group of World Wide Web Consortium (W3C)
  - Define standard

3

History

1988 SGML
Standard Generalized Markup Language
- Annotate text with structure

1992 HTML
Hypertext Mark-up Language
- Documents that are linked pieces
- Simple structure of language

1996 XML
- 1998 XML 1.0

4

XML

On surface looks much like HTML:

- Tags: <title> *title of document*</title>
- Structure: tags within tags
  <body><table> …</table> … </body>
  – Must be nested → hierarchy
- Tags have attributes <body bgcolor="#ffffff">

But tags are user-defined

- General *metadata*

5

XML

- Originally tags generalized description of document display– allow flexibility in markup
- Now tags can have *any* meaning
  – parties using *agree in advance* as to meaning
- Can use as data specification

XML has become major vehicle of exchanging data among unrelated, heterogeneous parties
  – Internet major vehicle of distribution

data-centric ⟷ text-centric
databases ⟷ information retrieval

6

Example XML: data-centric

```
<students>
   <student>
        <year>2007</year>
        <name><fn>Joe </fn><ln>Jones</ln></name>
        <address>…</address>
        <course type="deptal">cos 425</course>
        <course type="deptal">cos 432</course>
        <course type="elective">eng 331</course>
        etc.
   </student>
   <student> ……</student>
        ….
</students>
```

7

Example XML: mixed

Hamlet mark-up by Jon Bosak

will post xml file (read as plain text)

8

<SCENE><TITLE> SCENE III. A room in Polonius' house.</TITLE>
<STAGEDIR>Enter LAERTES and OPHELIA </STAGEDIR>

<SPEECH>
<SPEAKER>LAERTES</SPEAKER>
<LINE>My necessaries are embark'd: farewell:</LINE>
<LINE>And, sister, as the winds give benefit</LINE>
<LINE>And convoy is assistant, do not sleep,</LINE>
<LINE>But let me hear from you.</LINE>
</SPEECH>

<SPEECH>
<SPEAKER>OPHELIA</SPEAKER>
<LINE>Do you doubt that?</LINE>
</SPEECH>

Excerpt of marked-up play

9

Important XML concepts

- Information/data contained in a document
- Tags contain text and other tags
- Tags can be repeated arbitrary number of times
- Tags may or may not appear
  (Example next slide)
- Attributes of tags may or may not appear
  - attributes are strings
  - example <PLAY type="tragedy"> …
- Tags need not appear in rigid order

10

Example: tags may or may not appear

<SPEECH>
<SPEAKER>HAMLET</SPEAKER>
<LINE>Your loves, as mine to you: farewell.</LINE>
<STAGEDIR>Exeunt all but HAMLET**</STAGEDIR>**
<LINE>My father's spirit in arms! all is not well;</LINE>
<LINE>I doubt some foul play: would the night were come!</LINE>
<LINE>Till then sit still, my soul: foul deeds will rise,</LINE>
<LINE>Though all the earth o'erwhelm them, to men's eyes.</LINE>
</SPEECH>

11

Benefits of XML representation

- Self documenting by tag names
- Flexible formatting
  - Can introduce new tags or values
- Format can evolve without invalidating old
- Can have multi-valued components
  - e.g. courses of student, authors of book
- Wide variety of tools can process
  - Browsers
  - DB tools

12

Undesirable properties of XML representation

- Verbose representation:
  - repetition of tag names
    - Inefficient
- Redundant representation
  - Strict hierarchy
    - e.g. shared text in two sections of a document must be repeated

13

Specification

Need exchange syntax (semantics?) as well as XML document:

- XSL – eXtensible Style Language
  - How display information
- DTD = Document Type Declaration
  - User specifies own tags and attributes
  - User-defined grammar for syntax
- ➢ XML Schema
  - similar to but more general than DTD

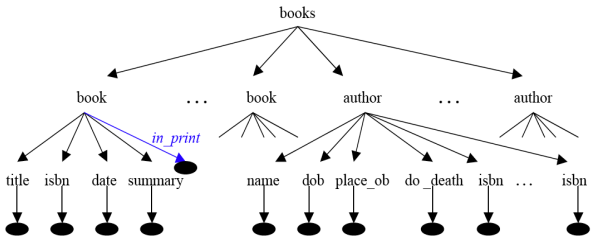
14

Semistructured Data Model

- XML gives structure, but not fully or rigidly specified
- Tag <> … </> defines XML element
  - Elements may contain sub-elements
  - Elements may contain values
  - Elements may have attributes
- Use labeled tree model (Document Object Model)
  - Element → node: atomic or compound object
  - Leaves: values and attributes

15

Graph model



XML Schema Example (simplified)

```
< xs:schema xmlns:xs="http://www.w3.org/2001/
   XMLSchema">
<xs:element name="books" type="ListBooksType"/>
<xs:element name="book" type="BookType"/>
<xs:element name="author" type="AuthorType"/>
<xs:complexType  name="ListBooksType">
        <xs:sequence>
            <xs:element ref="book" minOccurs="1"
              maxOccurs="unbounded"/>
            <xs:element ref="author" minOccurs="1"
              maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
```

17

example continued, 1

```
<xs:complexType  name="BookType">
        <xs:attribute name="in_print"/>
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="isbn" type="xs:string"/>
            <xs:element name="date" type="xs:string"/>
            <xs:element name="summary" type="xs:string"/
              >
        </xs:sequence>
</xs:complexType>
```

18

example continued, 2

```
<xs:complexType  name="AuthorType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="dob" type="xs:string"/>
            <xs:element name="place_ob" type="xs:string"/
              >
            <xs:element name="do_death" type="xs:string"/
              >
            <xs:element name= "isbn" type="xs:string"
              minOccurs= "0" maxOccurs=  "unbounded"/>
        </xs:sequence>
</xs:complexType>
</xs:schema>
```

19

XML Tools

- Display
  - Very flexible what and how display
- Convert to different representation
  - Example: put in relational database?
  - Example: build inverted index?
- Extract information from XML document
  - ➢Querying

20

Querying XML

- Storing data in XML; want to query
- Several querying languages
  - XPath : now building block
  - Quilt : historic
  - XQuery
  - XSLT : designed for style sheets but general
  - NEXI: extended Xpath
  - …

21

XQUERY

- Specified by W3C working group
  - Circa 2000
  - 2010 version 1.0
  - 2014 version 3.0
- Derived from older languages
- Modeled after SQL
  - data-centric
  - returns XML fragments
- Also useful for IR
  - Want, at minimum, path spec.
  - sometimes want attribute spec.

22

Path expression

- Traverse paths of tree
  - Use element names to name path
- Take ***all*** matching branches
- Returns sequence of nodes of tree
  - Node = XML elements

Doc. Identifier // element name /

e.g. URL root of tree — indicates element nested anywhere- jump down tree at this point in path — indicates immed. child of path so far

e.g. hamlet.xml/play//scene/title — title tag not only for scenes

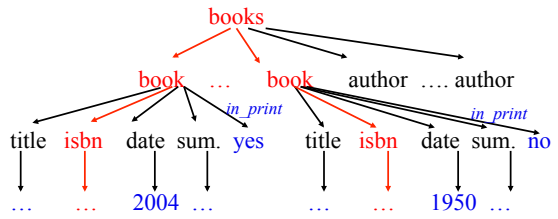
23

Path expressions – *some* details

- Returns sequence of matching elements
  - Includes tags of those elements
  - Sequence ordered by appearance in document
- Attributes can be accessed: @attribute_name
- … /* denotes *all children* of elements …/
- Predicates at any point in path
  - Prunes out paths
  - e.g. /students/student/course[@type= 'deptal']
- Doc(*document name*) returns root of a named document
  - File name
  - URL (URI)

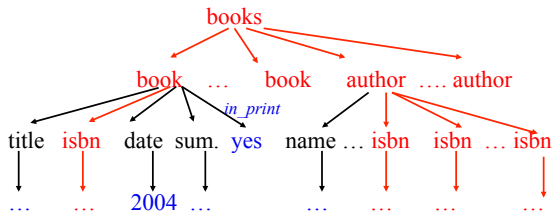
24

Data-centric example: /books/book/isbn



25

Data-centric example: /books//isbn



26

Xquery Example

FOR $x IN doc_id//name/ln
RETURN < LastName >{$x/text()}</LastName >

Gives: ?
For : <students>
<student>
<year>2007</year>
<name><fn>Joe </fn><ln>Jones</ln></name>
…
</student>
<student>
<year>2008</year>
<name><fn>Jane </fn><ln>Smith</ln></name>
…
</student>

</students>

27

Xquery features

- Example:
  FOR $x IN doc_id//name/ln
  RETURN < LastName >{$x/text()}</LastName >

  Gives: <LastName>Jones</LastName>
  < LastName >Smith</LastName >
- Returns XML fragments
- Many functions

28

What about information retrieval?

- How do we want to search an XML document with unstructured content?

29

Issues in XML text-centric retrieval

1. What is structure of document?
   - fine-grain structure
     - Shakespeare plays tagged to line
     - may want full path specification
     - simple search may suffice within text elements

hamlet.xml/play//scene [title has "woods"]//speech [speaker = "Hamlet"]

⇩

   - course-grain structure
     - entire body of document one text block
     - simple path specification
     - full IR search capability

books/book [body retrieve "science art"]

30

Issues in XML text-centric retrieval

2. How fine-grained does user want result?
   - document, section, paragraph, …
   - user interface to support path-based or schema-based queries?
3. How index document?
   - what parts of document indexed?
   - what is unit of document indexed?
     - know entire path of text element?
     - problems if too course-grained?
     - problems if too fine-grained?

31

Issues in XML text-centric retrieval

4. Heterogeneous or homogeneous collection
   - homogeneous: usually one (possibly distributed) source
     - e.g. Library of Congress
   - homogeneous: can have customized search interfaces
   - heterogeneous: many uncoordinated or loosely coordinated sources
     - e.g. Web
   - heterogeneous: schema may not be uniform
     - different labels
     - variations on structure

32

Other issues

- structural constraints as mandatory or hints?
- how structure affect ranking?
- removing redundancy due to results in nested elements

33