COS 597D:
Principles of
Database and Information Systems

XML and Information Exchange

XML
eXtensible Markup Language

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

Outline

- XML document structure
- XML querying with XQuery
- XML name spaces
- XML Schema definition

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*

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

Example XML

```
<students>
   <student>
        <startyear>2011</startyear>
        <name><fn>Joe </fn><ln>Jones</ln></name>
        <address>…</address>
        <course type="dept">cos 597A</course>
        <course type="dept">cos 402</course>
        <course type="elective">wri 503</course>
        etc.
   </student>
   <student> ……</student>
        ….
</students>
```

```
<books>
  <book in_print="true">
      < title> Of  Mice and Men< /title>
      <isbn>0140177396</isbn>
      <edition> 1 </edition>
      <publisher>Penguin</publisher
      <date>1993 </date>
  </book>
  …
  <book>…</book>
  <author>
      <name> John Steinbeck</name>
      <dob>1902 </dob>
      <do_death>1968</do_death>
      <isbn>0140177396 </isbn>
      …
      <isbn>014017737X </isbn>
  </author>
  …
  <author>…</author>
</books>
```

7

Important XML concepts

- Information/data contained in a document
  - Document = Database
- Tags contain text and other tags
- Tags can be repeated arbitrary number of times
- Tags may or may not appear
  - Example for <student>: …<generals>April 2011</generals>…
- Attributes of tags (strings) may or may not appear
- Tags need not appear in rigid order

8

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

9

Semistructured Data Model

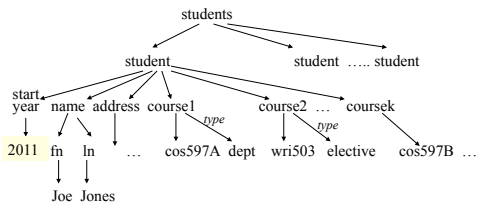
- XML gives structure, but not fully or rigidly specified
- Tag <> … </> defines XML element
  - Elements may contain sub-elements
  - Elements may contain values
  - Elements may have attributes
- Use labeled tree model
  - Element → node: atomic or compound object
  - Leaves: values and attributes

10

Example

```
<students>
  <student>
      <startyear>2011</startyear>
      <name><fn>Joe </fn><ln>Jones</ln></name>
      <address>…</address>
      <course type="dept">cos 597A</course>
      <course type="elective">wri 503 </course>
      etc.
  </student>
  <student> ……</student>
      ….
</students>
```

11



12

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

13

Undesirable properties of XML representation

- Verbose representation:
  repetition of tag names
  - Inefficient
- Redundant representation
  - conceptual directed acyclic graph gives duplicated elements
    - e.g. identical settings for two scenes of a play
      - same place, time of day, weather

14

XML Tools

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

15

Querying XML

- Storing data in XML; want to query
- Could map to relational model, but then must restructure data
- Several querying languages
  - XPath : now building block
  - Quilt : historic
  - XQuery
  - XSLT : designed for style sheets but general

16

XQUERY

- Specified by W3C working group
  - Circa 2000
  - Latest XQuery 1.1 Dec. 2010
- Derived from older languages
- Modeled after SQL

17

Brief look at XQUERY

FLWOR (flower) expression:

- FOR *path expression* – anal. to SQL "FROM"
- LET *variable name* = *path expression* – anal. To SQL "AS"
- WHERE *condition* – anal. to SQL "WHERE"
- ORDER BY – anal. to SQL "ORDER BY"
- RETURN – constructs XML result – anal to SQL "SELECT"

XQUERY returns XML fragment

- XML $\xrightarrow{\text{XQuery}}$ XML
  - Compare: relations $\xrightarrow{\text{SQL}}$ relation

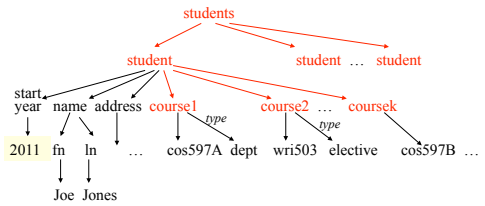
18

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. /students/student/course



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='dept']
- Doc(*document name*) returns root of a named document
  - File name
  - URL (URI)

XQuery FOR …

For $x in *path expression 1,*
$y *in path expression 2,*
…

- $ precedes variable name
- Each variable ranges over sequence of elements returned by its path expression
- Multiple variables => Cartesian product

XQuery Let …

Let $z := *path expression1*
Let $q := *path expression2*
…

Value of variable (e.g. $z) is entire sequence if path expression returns sequence

XQuery WHERE …

WHERE *predicate*

- Predicate on set defined in FOR
  FOR $b IN /students/student
  WHERE $b/startyear='2011'
- Rich set of functions, comparison operations

XQuery RETURN …

- Constructs XML result
- Give explicit tags for result
- Give expressions to be evaluated
  {*expression*}
- Example

```
FOR $b IN doc_id/students/student
WHERE $b/startyear='2013'
RETURN <Result>{$b/name/fn $b/name/ln} </Result>
```

Gives: <Result><fn>Joe</fn><ln><Jones></ln></Result>
<Result> …
etc.

25

Example

```
FOR $x IN doc_id//name/ln
RETURN <LastName>{$x}</LastName>
```

Gives: ?
For :

```
<students>
  <student>
      <startyear>2013</startyear>
      <name><fn>Joe </fn><ln>Jones</ln></name>
      …
  </student>
  <student>
      <startyear>2010</startyear>
      <name><fn>Jane </fn><ln>Smith</ln></name>
      …
  </student>

</students>
```

26

Examples

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

Gives: <LastName><ln>Jones</ln></LastName>
< LastName ><ln>Smith</ln></LastName >

27

Examples

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

Gives: <LastName>Jones</LastName>
< LastName >Smith</LastName >

- Many functions

28

FOR versus LET

```
for $title in /BOOKS/BOOK/TITLE
return <TITLES>{$title}</TITLES>

gives:
<TITLES>
    <TITLE>The Tragedy of Romeo
      and Juliet</TITLE>
</TITLES>

<TITLES>
    <TITLE>The Tragedy of Hamlet,
      Prince of Denmark</TITLE>
</TITLES>

<TITLES>
    <TITLE>The Tragedy of
      Macbeth</TITLE>
</TITLES>
```

```
let $title := /BOOKS/BOOK/TITLE
return <TITLES>{$title}</TITLES>

gives:
<TITLES>
    <TITLE>The Tragedy of Romeo
      and Juliet</TITLE>
    <TITLE>The Tragedy of Hamlet,
      Prince of Denmark</TITLE>
    <TITLE>The Tragedy of
      Macbeth</TITLE>
</TITLES>
```

XQuery: A *very* incomplete list of features

- Are aggregation operations
- Can nest XQuery expressions in RETURN clause
  - Can get nested elements in result not nested in original
- Get joins: conditions in WHERE coordinate paths expressions over variables in FOR
- Can have if…then ...else within RETURN clause
- Can have quantification within WHERE clause
  - SOME $e IN *path expression* SATISFIES *predicate with $e free*
  - EVERY $e IN …

30

Outline

- ✓ XML document structure
- ✓ XML querying with XQuery
- • XML name spaces
- • XML Schema definition

31

Namespaces

- Exchanging XML documents with unrelated sites, unrelated applications requires unambiguous identifiers across sources of documents
- XML allows each source to specify a globally unique name: universal resource identifiers (URIs)
  - URLs
- Names within one source expect source to keep unambiguous

32

Namespace specification

- Prepend URI to each tag or attribute name
  http://www.princeton.edu:student
- Verbose – have abbreviation mechanism
  Attribute within *root* tag: xmlns:*abbrev*="*URI*"

```
<students xmlns:PUstu="http://www.princeton.edu">
    <PUstu:student>
    <PUstu:year>2005</PUstu:year>
    …
```

Becomes part of tag name

33

Multiple namespaces

- One document can have several namespaces defined and used
  - Different sources
  - Sources need not be sites
- Namespace can denote specific XML standard
  - Extend types
  - Extend functions

xmlns:xs="http://www.w3.org/2001/XMLSchema"
Get types "xs:string", "xs:decimal"

Leads us to …

34

Outline

- ✓ XML document structure
- ✓ XML querying with XQuery
- ✓ XML name spaces
- • **XML Schema definition**

35

Language ***XML Schema***

Standard for specifying schema of a document:

- Specify tag names, attribute names
- Declare leaf types (contents)
  - Built-in types
  - User-defined types
- Declare tag structure
  - tree model
- Specify constraints:
  - key
  - foreign key

36

XML Schema specification

The schema for a document is
an XML document

Says using specific w3c standard from namespace

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

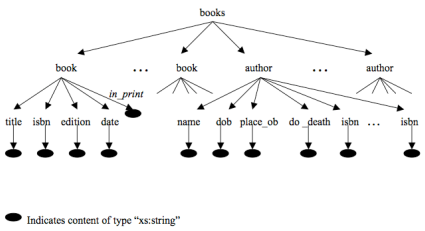
. . . *specification of document*

</xs:schema>

37

Example for Schema definition

Gives schema for tree model:



Indicates content of type "xs:string"

38

XSchema Basics

- Declare elements (nodes of tree)

<xs:element name="…" type="…" > … </xs:element>

name of element declaring — type of element — content: nested elements attributes

– if no nested elements and element has no attributes, can abbreviate to <xs:element name="…" type="…" />

example
<xs:element name="isbn" type="xs:string" />

39

Nested elements

Define named complex type

<xs:complexType name="*typename*">
<xs:sequence>
<xs: element name="…" …>
</xs:element>
…
<xs: element name="…" …>
</xs:element>
</xs:sequence>
</xs:complexType>

declaration for nested elements

<xs:element name="…" type= "*typename*"/>

could be nesting within nesting

40

<xs:element name="**book**" type="BookType"/>

<xs:complexType name="BookType">
<xs:sequence>
<xs:element name="**title**" type="xs:string"/>
<xs:element name="**isbn**" type="xs:string"/>
<xs:element name="**edition**" type="xs:string"/>
<xs:element name="**date**" type="xs:string"/>
</xs:sequence>

</xs:complexType>

42

<xs:element name="**author**" type="AuthorType"/>

<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>

43

Other parts specification

- attribute declaration: in content part:
  <xs:attribute name="…" type="…" />
- refer to previously defined element:
  <xs:element ref="*name of prev. defined element*" />
- multiple occurrences of element in a sequence
  - specify and quantify

```
<xs:sequence>
    <xs:element … minOccurs="…" maxOccurs="…" >
</xs:sequence>
```

44

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

45

```
<xs:element name="books" type="ListBooksType"/>

<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>
```

46

Primary keys and Foreign keys

- defining a candidate key:

```
<xs:key name="name you give" >
    <xs:selector xpath=" a path specification " />
    <xs:field xpath = "names of fields" />
</xs:key>
```

path to key

elements and attributes that make up key

- defining a foreign key constraint:

```
<xs:keyref name="name you give"
        refer="name of candidate key referencing">
    <xs:selector xpath=" a path specification " />
    <xs:field xpath = "names of fields" />
</xs:keyref>
```

- These top-level definitions within scheme

47

```
<xs:key name= "bookKey" >
    <xs:selector xpath= "/books/book"/>
    <xs:field xpath= "isbn"/>
</xs:key>

<xs:keyref name= "authorBookFkey" refer= "bookKey" >
    <xs:selector xpath= "/books/author"/>
    <xs:field xpath= "isbn"/>
</xs:keyref>
```

48

Putting example all together

```
< xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="books" type="ListBooksType"/>
    <xs:element name="author" type="AuthorType"/>
    <xs:element name="book" type="BookType"/>
    <xs:complexType name="AuthorType">
        …
    </xs:complexType>
    <xs:complexType name="BookType">
        …
    </xs:complexType>
    <xs:complexType name="ListBooksType">
        …
    </xs:complexType>
    <xs:key name= "bookKey" > … </xs:key>
    <xs:keyref name= "authorBookFkey" refer= "bookKey" > … </xs:keyref>
</xs:schema>
```

49

XML for information exchange

Many and wide range of applications use XML to exchange information (data). Some examples:

- PADS tool here (Prof. Walker) converts "ad hoc" (nonstandard) data file into an XML file
  - XML one of choices
- XML standards for specifying 3D models
  - U3D (Adobe Acrobat)
  - Collada (Google Earth)
- describe security vulnerabilities
- W3C specify XML standards
- Digital humanities text studies
  - "Advancing XML-Based Scholarship from Representation to Discovery: The XQuery Summer Institute at Vanderbilt University is aimed at archivists, librarians, professors, and students who have experience marking up texts in XML, but do not yet know how to work computationally with those documents."

50