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>  ... <p>  ...</p> </body>
  - Must be nested — hierarchy
• Tags have attributes <body bgcolor="#f0f0f0">

But Tags are User-defined
• General metadata

Example XML

@student>
  <startyear>2008</startyear>
  <name>Joe Jones</name>
  <address>...</address>
</student>

<course type="dept">cos 597A</course>
<course type="dept">cos 402</course>
<course type="elective">wri 503</course>

etc.
</student>

@student> .... </student>
...
</students>
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 2008</generals>…
• Attributes of tags (strings) may or may not appear
• Tags need not appear in rigid order

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

Undesirable properties of XML representation

• Verbose representation:
  repetition of tag names
  • Inefficient
• Redundant representation
  – Document contains all info, even if much does not change
  • e.g. document containing employee info:
    basic name, address, etc. repeated even if only assignment changes
  • Compare one table in relational DB

Board Example

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

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
Example

```xml
<students>
  <student>
    <startyear>2008</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>
```

XML Tools

- Display
  - Very flexible what and how display

- Convert to different representation
  - Example: put in relational database?

- Extract information from XML document
  - Querying

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

XQUERY

- Specified by W3C working group
  - Circa 2000
- Derived from older languages
- Modeled after SQL

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 XQuery XML SQL relation
- Compare: relations relation
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 indicates element indicates immed.
root of tree nested anywhere child of path so for
jump down tree at this point in path

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,
...

- $x$ 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='2009'
- 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='2009'
  RETURN \{Result\}{$b$/name/fn $b$/name/ln} \</Result\>
  Gives:
  \{Result\}<fn>Joe</fn><ln>Jones</ln> \</Result\>
  etc.

Example

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

Gives:  
 For:  
  <student>
    <student>
      <name><fn>Joe</fn><ln>Jones</ln></name>
      ...
    </student>
  </student>

Examples

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

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

• Many functions

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 ...

Outline

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

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="URF"`

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

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

```xml
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

Get types “xs:string”, “xs:decimal”

Leads us to ...

Outline

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

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

**XML Schema** specification

The schema for a document is an XML document:

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
  specification of document
</xs:schema>
```
### XScheme Basics

- **Declare elements (nodes of tree)**
  ```xml
  <xs:element name="..." type="..."/>
  ```
  - 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**
  ```xml
  <xs:element name="isbn" type="xs:string"/>
  ```

### Nested elements

- **Choice 1:**
  ```xml
  <xs:complexType>
    <xs:sequence>
      <xs:element name="..." type="..."/>
    </xs:sequence>
  </xs:complexType>
  ```
  - Could be nesting within nesting declaration for nested elements

### Define named complex type

- **Choice 2:**
  ```xml
  <xs:complexType name="typename">
    <xs:sequence>
      <xs:element name="..." type="..."/>
    </xs:sequence>
  </xs:complexType>
  ```

### Other parts specification

- **Attribute declaration:** in content part:
  ```xml
  <xs:attribute name="..." type="..."/>
  ```
  - Refer to previously defined element:
    ```xml
    <xs:element ref="name of prev. defined element"/>
    ```
  - Multiple occurrences of element in a sequence
    - Specify and quantify
      ```xml
      <xs:sequence>
        <xs:element ... minOccurs="..." maxOccurs="..."/>
      </xs:sequence>
      ```

### Primary keys and Foreign keys

- **Defining a candidate key:**
  ```xml
  <xs:element name="..." type="..."/>
  ```
  - Path to key
    ```xml
    <xs:selector xpath="a path specification"/>
    ```
  - Elements and attributes that make up key
    ```xml
    <xs:field xpath="names of fields"/>
    ```

- **Defining a foreign key constraint:**
  ```xml
  <xs:keyref name="..." key="..."/>
  ```
  - Referencing a candidate key referencing
    ```xml
    <xs:selector xpath="a path specification"/>
    ```
  - Elements and attributes that make up key
    ```xml
    <xs:field xpath="names of fields"/>
    ```

- **These top-level definitions within scheme**
XML uses 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
    - Acrobat (U3D)
    - Google (Collada)
  - Describe security vulnerabilities
  - W3C specify XML standards

SUMMARY

- XML is language for representing information (data) in semi-structured way
  - Self documenting by tag names
  - Flexible formatting
  - Began as language for generalizing specification of document display
- Generality allows XML to be important information exchange format for Internet
- **XML Schema** provides formal specification of document schema
- **XQuery** provides SQL-like query language for extracting information from an XML document