Lecture 12:
XML, XPath, XQuery

Friday, October 25, 2002

Outline

- XML: syntax, semantics, data, DTDs
- XPath
- XQuery

Strongly recommended readings:

- XPath:
  http://java.sun.com/webservices/docs/ea2/tutorial/doc/JAXPXSLT2.html
- XQuery:
  http://www.w3.org/TR/xmlquery-use-cases/
  http://www.xmlportfolio.com/xquery.html

XML Syntax

- 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: if it has matching tags

XML Syntax

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

attributes are alternative ways to represent data

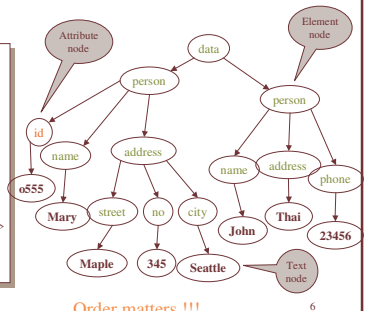
XML Syntax

```
<person id="o555">  <name> Jane </name> </person>

<person id="o456">  <name> Mary </name>
                    <children idref="o123 o555"/>
</person>

<person id="o123" mother="o456"><name>John</name>
</person>
```

oids and references in XML are just syntax

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
  - Reational schema: persons(name,phone)
  - In XML <persons>, <name>, <phone> are part of the data, and are repeated many times
- Consequence: XML is much more flexible
- XML = semistructured data

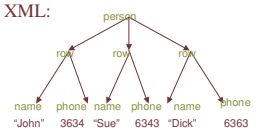
7

Relational Data as XML

person

name	phone
John	3634
Sue	6343
Dick	6363

XML:




```
<person>
   <row> <name>John</name>
         <phone> 3634</phone></row>
   <row> <name>Sue</name>
         <phone> 6343</phone>
   <row> <name>Dick</name>
         <phone> 6363</phone></row>
</person>
```

XML is Semi-structured Data

- 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	-

9

XML is Semi-structured Data

- Repeated attributes

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

← two phones !

- Impossible in tables:

name	phone		
Mary	2345	3456	???

10

XML is Semi-structured Data

- Attributes with different types in different objects

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

← structured name !

- Nested collections (no 1NF)
- Heterogeneous collections:
  - <db> contains both <book>s and <publisher>s

11

Document Type Definitions DTD

- part of the original XML specification
- 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

12

Very Simple 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)>
]>
```

13

Very Simple 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>
```

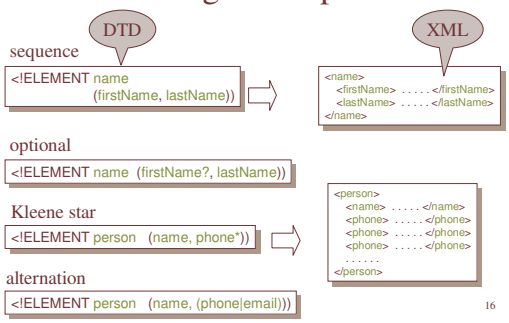
14

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)*

15

DTD: Regular Expressions



16

Querying XML Data

- XPath = simple navigation through the tree
- XQuery = the SQL of XML
- XSLT = recursive traversal
  - will not discuss in class

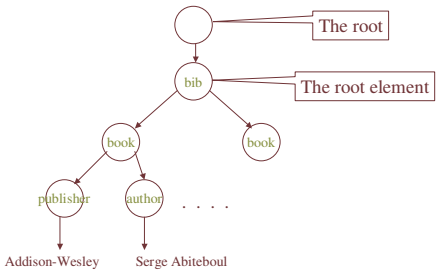
17

Sample Data for Queries

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                     <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
</bib>
```

18

Data Model for XPath



19

XPath: Simple Expressions

/bib/book/year

Result: <year> 1995 </year>
<year> 1998 </year>

/bib/paper/year

Result: empty (there were no papers)

20

XPath: Restricted Kleene Closure

//author

Result: <author> Serge Abiteboul </author>
<author> <first-name> Rick </first-name>
<last-name> Hull </last-name>
</author>
<author> Victor Vianu </author>
<author> Jeffrey D. Ullman </author>

/bib//first-name

Result: <first-name> Rick </first-name>

21

Xpath: Text Nodes

/bib/book/author/text()

Result: Serge Abiteboul
Jeffrey D. Ullman

Rick Hull doesn't appear because he has firstname, lastname

Functions in XPath:

- text() = matches the text value
- node() = matches any node (= * or @* or text())
- name() = returns the name of the current tag

22

Xpath: Wildcard

//author/*

Result: <first-name> Rick </first-name>
<last-name> Hull </last-name>

* Matches any element

23

Xpath: Attribute Nodes

/bib/book/@price

Result: "55"

@price means that price is has to be an attribute

24

Xpath: Predicates

/bib/book/author[firstname]

Result: <author> <first-name> Rick </first-name>
<last-name> Hull </last-name>
</author>

25

Xpath: More Predicates

/bib/book/author[firstname][address[//zip][city]]/lastname

Result: <lastname> … </lastname>
<lastname> … </lastname>

26

Xpath: More Predicates

/bib/book[@price < “60”]

/bib/book[author/@age < “25”]

/bib/book[author/text()]

27

Xpath: Summary

bib	matches a bib element
*	matches any element
/	matches the root element
/bib	matches a bib element under root
bib/paper	matches a paper in bib
bib//paper	matches a paper in bib, at any depth
//paper	matches a paper at any depth
paper|book	matches a paper or a book
@price	matches a price attribute
bib/book/@price	matches price attribute in book, in bib
bib/book/[@price<“55”]/author/lastname	matches…

28

XQuery

- Based on Quilt, which is based on XML-QL
- Uses XPath to express more complex queries

29

FLWR (“Flower”) Expressions

FOR ...
LET...
WHERE...
RETURN...

30

XQuery

Find all book titles published after 1995:

```
FOR $x IN document("bib.xml")/bib/book
WHERE $x/year > 1995
RETURN { $x/title }
```

Result:

```
<title> abc </title>
<title> def </title>
<title> ghi </title>
```

31

XQuery

Find book titles by the coauthors of "Database Theory":

```
FOR $x IN bib/book[title/text() = "Database Theory"]/author
    $y IN bib/book[author/text() = $x/text()]/title
RETURN <answer> { $y/text() } </answer>
```

The answer will contain duplicates !

Result:

```
<answer> abc </ answer >
< answer > def </ answer >
< answer > ghi </ answer >
```

32

XQuery

Same as before, but eliminate duplicates:

```
FOR $x IN bib/book[title/text() = "Database Theory"]/author
    $y IN distinct(bib/book[author/text() = $x/text()]/title)
RETURN <answer> { $y/text() } </answer>
```

distinct = a function that eliminates duplicates

Result:

```
<answer> abc </ answer >
< answer > def </ answer >
< answer > ghi </ answer >
```

33

XQuery: Nesting

For each author of a book by Morgan Kaufmann, list all books she published:

```
FOR $a IN distinct(document("bib.xml")
                   /bib/book[publisher="Morgan Kaufmann"]/author)
RETURN <result>
          { $a,
            FOR $t IN /bib/book[author=$a]/title
            RETURN $t
          }
       </result>
```

34

XQuery

Result:

```
<result>
      <author>Jones</author>
      <title> abc </title>
      <title> def </title>
  </result>
  <result>
       <author> Smith </author>
       <title> ghi </title>
  </result>
```

35

XQuery

- FOR $x in expr -- binds $x to each value in the list expr
- LET $x = expr -- binds $x to the entire list expr
  - Useful for common subexpressions and for aggregations

36

XQuery

```
<big-publishers>
  FOR $p IN DISTINCT-DOCUMENT("bib.xml")//publisher
  LET $b := DOCUMENT("bib.xml")/book[publisher = $p]
  WHERE count($b) > 100
  RETURN <big> $p </big>
</big-publishers>
```

count = a (aggregate) function that returns the number of elms

XQuery

Find books whose price is larger than average:

```
LET $a=avg(DOCUMENT("bib.xml")/bib/book/price)
FOR $b in DOCUMENT("bib.xml")/bib/book
WHERE $b/price > $a
RETURN { $b }
```


XQuery

Summary:

- FOR-LET-WHERE-RETURN = FLWR

Instance of Xquery data model

FOR v.s. LET

FOR

- Binds *node variables* → iteration

LET

- Binds *collection variables* → one value

FOR v.s. LET

```
FOR $x IN DOCUMENT("bib.xml")/bib/book
RETURN <result> { $x } </result>
```

Returns:

```
<result> <book>...</book></result>
<result> <book>...</book></result>
<result> <book>...</book></result>
...
```

```
LET $x IN DOCUMENT("bib.xml")/bib/book
RETURN <result> { $x } </result>
```

Returns:

```
<result> <book>...</book>
         <book>...</book>
         <book>...</book>
         ...
</result>
```


Collections in XQuery

- Ordered and unordered collections
  - /bib/book/author = an ordered collection
  - Distinct(/bib/book/author) = an unordered collection
- LET $a = /bib/book → $a is a collection
- $b/author → a collection (several authors...)

```
RETURN <result> { $b/author } </result>
```

Returns:

```
<result> <author>...</author>
         <author>...</author>
         <author>...</author>
         ...
</result>
```

Collections in XQuery

What about collections in expressions ?

- $b/price → list of n prices
- $b/price * 0.7 → list of n numbers
- $b/price * $b/quantity → list of n x m numbers ??
- $b/price * ($b/quant1 + $b/quant2) ≠ $b/price * $b/quant1 + $b/price * $b/quant2 !!

43

Sorting in XQuery

```
<publisher_list>
  FOR $p IN distinct(document("bib.xml")//publisher)
  RETURN <publisher> <name> { $p/text() } </name> ,
              FOR $b IN document("bib.xml")//book[publisher = $p]
              RETURN <book>
                          { $b/title ,
                            $b/price
                          }
                     </book> SORTBY(price DESCENDING)
         </publisher> SORTBY(name)
</publisher_list>
```

44

If-Then-Else

```
FOR $h IN //holding
RETURN <holding>
            { $h/title,
              IF $h/@type = "Journal"
                    THEN $h/editor
              ELSE $h/author
            }
       </holding> SORTBY (title)
```

45

Existential Quantifiers

```
FOR $b IN //book
WHERE SOME $p IN $b//para SATISFIES
            contains($p, "sailing")
            AND contains($p, "windsurfing")
RETURN { $b/title }
```

46

Universal Quantifiers

```
FOR $b IN //book
WHERE EVERY $p IN $b//para SATISFIES
            contains($p, "sailing")
RETURN { $b/title }
```

47

Other Stuff in XQuery

- BEFORE and AFTER
  - for dealing with order in the input
- FILTER
  - deletes some edges in the result tree
- Recursive functions
  - Currently: arbitrary recursion
  - Perhaps more restrictions in the future ?

48