Lecture 6: XML Query Languages

Thursday, January 18, 2001

1

Outline

- XPath
- XML-QL
- XSL (XSLT)

2

An Example of XML Data

```
<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>
```

3

XPath

- Syntax for XML document navigation and node selection
- A *recommendation* of the W3C (i.e. a standard)
- Building block for other W3C standards:
  - XSL Transformations (XSLT)
  - XML Link (XLink)
  - XML Pointer (XPointer)
- Was originally part of XSL – "XSL pattern language"

4

XPath: Simple Expressions

/bib/book/year

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

/bib/paper/year

Result: empty (there were no papers)

5

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>

6

Xpath: Text Nodes

/bib/book/author/text()

Result: Serge Abiteboul
Jeffrey D. Ullman

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

7

Xpath: Wildcard

//author/*

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

* Matches any element

8

Xpath: Attribute Nodes

/bib/book/@price

Result: "55"

@price means that price is has to be an attribute

9

Xpath: Qualifiers

/bib/book/author[firstname]

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

10

Xpath: More Qualifiers

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

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

11

Xpath: More Qualifiers

/bib/book[@price < "60"]

/bib/book[author/@age < "25"]

/bib/book[author/text()]

12

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…

13

Xpath: More Details

- An Xpath expression, p, establishes a relation between:
  - A *context node*, and
  - A node in the *answer set*
- In other words, p denotes a function:
  - S[p] : Nodes -> {Nodes}
- Examples:
  - author/firstname
  - . = self
  - .. = parent
  - part/*/*/subpart/../name = what does it mean ?

14

The Root and the Root

- <bib> <paper> 1 </paper> <paper> 2 </paper> </bib>
- bib is the "***document element***"
- The "***root***" is above bib

- /bib = returns the document element
- / = returns the root

- Why ? Because we may have comments before and after <bib>; they become siblings of <bib>

- This is advanced xmlogy

15

Xpath: More Details

- We can navigate along 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

16

Xpath: More Details

- Examples:
  - child::author/child:lastname = author/lastname
  - child::author/descendant::zip = author//zip
  - child::author/parent::* = author/..
  - child::author/attribute::age = author/@age

17

XML-QL: A Query Language for XML

- http://www.w3.org/TR/NOTE-xml-ql (8/98)
- features:
  - regular path expressions
  - patterns, templates
  - subqueries
  - Skolem Functions
- based on a graph model (the OEM data model)
  - sometimes things don't work smoothly with XML

18

Pattern Matching in XML-QL

```
where <book language="french">
            <publisher>
                  <name> Morgan Kaufmann </name>
            </publisher>
            <author> $a </author>
          </book> in "www.a.b.c/bib.xml"
construct $a
```

<book …> … </book> is called a ***pattern***
Pattern = like XML fragment, but may have variables

Abbreviations in XML-QL

```
where <book language="french">
            <publisher>
                  <name> Morgan Kaufmann </name>
            </>
            <author> $a </>
          </> in "www.a.b.c/bib.xml"
construct $a
```

</element> abbreviated with </>

Simple Constructors in XML-QL

```
where <book language = $l>
                <author> $a </>
          </> in "www.a.b.c/bib.xml"
construct <result> <author> $a </> <lang> $l </> </>
```

<result>…</> is called a ***template***

Answer is:

```
<result> <author>Smith</author> <lang>English   </lang></result>
<result> <author>Smith</author> <lang>Mandarin</lang></result>
<result> <author>Doe   </author> <lang>English   </lang></result>
```


Regular Expressions in XML-QL

- Uses traditional syntax for regular expressions

```
where <product.(part)*.subpart?>
            <description>
                  <name|nome>  spring </>
                  <manufacturer>$m</>
            </>
            <price> $p </>
          </book> in "www.a.b.c/products.xml"
construct <result><man>$m</> <cost>$p</></>
```


Regular Expressions in XML-QL

- Can use the following:

  R ::= tag | _ | R.R | R|R | R* | R+ | R?

- Notice: XPath corresponds to:

  R ::= tag | _ | R.R | R|R | _*

Nested Queries in XML-QL

```
where <bib.paper.author> $a </>
            in "www.a.b.c/bib.xml"
construct <author>
                  <name> $a </>
                  where <bib.paper>
                              <author> $a </>
                              <title> $t </>
                        </> in "www.a.b.c/bib.xml"
                  construct <title> $t </>
            </>
```

Nested Queries in XML-QL

- Results will be grouped by authors:

```
<author> <name> John </name>
         <title> t1 </title>
         <title> t2 </title>
         …
</author>
<author> <name> Smith </name>
         <title> … </title>
         …
</author>…
```

- What happens to duplicate authors ? Need Skolem functions…

Representing References in XML

```
<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>
```

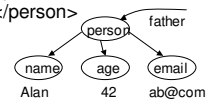
oids and references in XML are just syntax

Note: References in XML vs Semistructured Data

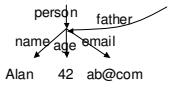
```
<person id="o123">
  <name> Alan </name>
  <age> 42 </age>
  <email> ab@com </email>
</person>
```

```
{ person: &o123
    { name: "Alan",
      age: 42,
      email: "ab@com" }
}
```

```
<person father="o123"> …
</person>
```




```
{ person: { father: &o123 …}
}
```



similar on trees, different on graphs

Skolem Functions in XML-QL

```
where <bib.book>
        <author> $a </>
        <title> $t </>
      </> in "www.a.b.c/bib.xml"
construct <result> <author id=F($a)> $a</>
                   <title> $t </>
            </>
```

What happens to duplicate authors ?

More on Skolem Functions

```
where <bib.book>
        <author> $a </>
        <title> $t </>
      </> in "www.a.b.c/bib.xml"
construct <result id=F($t)> <author id=G($a,$t)> $a</>
                            <title id=H($t)> $t </>
            </>
```

- what does it do ?
- what about the order ?

More on Skolem Functions

```
where <bib.book>
        <author> $a </>
        <title> $t </>
      </> in "www.a.b.c/bib.xml"
construct <result id=F($a,$t)> <author id=G($a)> $a</>
                               <title id=H($t)> $t </>
            </>
```

- what happens here ?
- need discipline in using Skolem functions, otherwise we get a graph

XSL

- = XSLT + XPath
- A recommendation of the W3C (standard)
- Initial goal: translate XML to HTML
- Became: translate XML to XML
  - HTML is just a particular case of XML

31

XSL Templates and Rules

- query = collection of template rules
- template rule = match pattern + template

Retrieve all book titles:

```
<xsl:template> <xsl:apply-templates/> </xsl:template>

<xsl:template match = "/bib/*/title">
      <result>  <xsl:value-of/>  </result>
</xsl:template>
```

32

XSL for Stylesheets

- Authors in italic, title in boldface

```
<xsl:template> <xsl:apply-templates/> </xsl:template>

<xsl:template match = "/bib">
    <h1>  All books in our database </h1>
    <xsl:apply-templates/>
</xsl:template>

<xsl:template match = "/bib/book/author">
      <result>  <i> <xsl:value-of/> </i>,  </result>
</xsl:template>

<xsl:template match = "/bib/book/title">
      <result>  <b> <xsl:value-of/> </b> <br/></result>
</xsl:template>
```

33

Input XML

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> Rick  Hull </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>
```

34

Output HTML

```
<h1>  All books in our database </h1>
<i> Serge Abiteboul </i>,
<i> Rick Hull </i>,
<i> Victor Vianu </i>,
 <b> Foundations of Databases </b>
</br>
<i>Jeffrey D. Ullman </i>,
<b> Principles of Database and Knowledge Base Systems </b>
<br/>
```

35

Flow Control in XSL

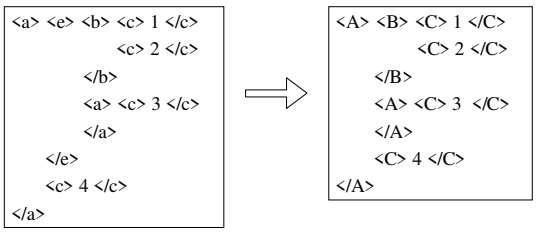
```
<xsl:template> <xsl:apply-templates/> </xsl:template>

<xsl:template match="a">  <A><xsl:apply-templates/></A>
</xsl:template>

<xsl:template match="b">  <B><xsl:apply-templates/></B>
</xsl:template>

<xsl:template match="c">  <C><xsl:value-of/></C>
</xsl:template>
```

36



37

XSLT

```
<xsl:template> <xsl:apply-templates/> </xsl:template>

<xsl:template match="a"> <a><xsl:apply-templates/></a>
                         <a><xsl:apply-templates/></a>
</xsl:template>
```

38

XSLT

- What is the output on:

<a> <a> <a>

?

39

- Answer:

40