Fall, 2007 CIS 550

Database and Information Systems

Solutions to Homework 4

Due on November 9, 2007

For this question, you should test your answers using SAXON, an XSLT and XQuery processor. If you would like to work from your own machine, you may download SAXON and its documentation from http://saxon.sourceforge.net/. Alternatively, you can ssh to *eniac.seas.upenn.edu* and use the resources in *~zives/saxon/* to test your query source file. You might need to add some lines to your .profile file first:

```
export JAVA_HOME=/usr/java/jdk1.6.0_02
export PATH=${PATH}:${JAVA_HOME}/bin
export CLASSPATH=${CLASSPATH}:~zives/saxon/saxon8.jar
```

Source your .profile. (This assumes your shell is bash. If not, simply type "bash" and then "source profile.") There's an example XQuery file "*example.xq*," which uses the XML data file "*book.xml*". Type in the command:

```
java net.sf.saxon.Query -t -s book.xml example.xq > example.xml
```

and view the result in example.xml

The XML data files used for this problem are **proc.xml** (list of conference proceedings) and **inproc.xml** (list of articles in proceedings) in *~zives/saxon/* on eniac. Based on the two XML files, write the following queries in XQuery, with the query output embedded within tags ⟨answer⟩ . . . ⟨/answer⟩. To test, you may type in the command:

```
java net.sf.saxon.Query -t -s proc.xml -s inproc.xml
  /yourdirectory/hw4_1_1.xq   >/yourdirectory/hw4_1_1.xml
```

And view your results in the hw4_1_1.xml file in your directory.

1. For each paper authored by Jeffrey D. Ullman, return a **paper** tag, with the following contents: the paper title (in the appropriate element) and Ullman's **coauthors**. Note that Ullman is not his own coauthor (coathorship is not reflexive).

```
<answer> {
   for $p in doc("inproc.xml")/dblp/inproceedings
   where $p/author="Jeffrey D. Ullman"
   return <paper>
            <title>{$p/title/text()}</title>
             { for $a in $p/author
               where $a!= "Jeffrey D. Ullman"
               return <coauthor>{$a/text()}</coauthor> }
         </paper>
} </answer>
```

2. Find the **authors** whose papers were most cited in the 1990s. Note that you can test the crossref to the proceedings key attribute (which ends with the year) to determine the publication year. The result should be nested in an **author** tag.

```
<answer> {
    let $maxcite := fn:max(
        for $p in doc('inproc.xml')/dblp/inproceedings
            let $c:= doc('inproc.xml')/dblp/inproceedings
            [cite=$p/@key and substring(year,1,3)="199"]
        return fn:count($c)
        )

    let $authors:=  fn:distinct-values(
        for $p in doc('inproc.xml')/dblp/inproceedings
            let $c:= doc('inproc.xml')/dblp/inproceedings
            [cite=$p/@key and substring(year,1,3)="199"]
        where fn:count($c)= $maxcite
        return $p/author
        )

    for $a in $authors
    return <author> {$a} </author> }
</answer>

XML result: <?xml version="1.0" encoding="UTF-8"?>
<answer>
   <author>Patricia G. Selinger</author>
   <author>Morton M. Astrahan</author>
   <author>Donald D. Chamberlin</author>
   <author>Raymond A. Lorie</author>
   <author>Thomas G. Price</author>
</answer>
```

3. Find the **authors** who published the most papers in the proceedings of SIGMOD (any

year). You may wish to use the XQuery function fn:contains() to match substrings. (See http://saxon.sourceforge.net/saxon7.6/functions.html.)

```
<answer> {
    let $count :=
        for $p in fn:distinct-values(doc("inproc.xml")/dblp/inproceedings/author)
            let $c:= doc("inproc.xml")/dblp/inproceedings
             [author=$p and fn:contains(@key, "sigmod")]
        return <pair>
                <author>{$p}</author>
                <count>{fn:count($c)}</count>
               </pair>
    return <author>{
            for $a in $count
            where $a/count = fn:max($count/count)
            return $a/author/text()
           }</author>
}</answer>

XML result:<?xml version="1.0" encoding="UTF-8" ?>
- <answer>
  <author>Michael J. Carey</author>
  </answer>
```

4. Return, for each paper in the proceedings of VLDB 2003, the **title** and the *second* **author**. If the paper only has one author, omit the **author** tag entirely (but still return the title). The result should be nested within a **paper** tag.

```
<answer> {
    for $p in doc("inproc.xml")/dblp/inproceedings
      [fn:contains(@key, "vldb") and year= "2003"]
    return <paper>
            <title>{$p/title/text()}</title>
             { for $a in $p/author[fn:position()=2]
               return <coauthor>{$a/text()}</coauthor> }
           </paper>
} </answer>
```

5. Find the distinct **titles** of proceedings in **proc** that don't have any paper information in **inproc**. The result should be nested in **title** tag.

```
<answer> {
    for $p in doc("proc.xml")/dblp/proceedings
    where fn:empty(doc("inproc.xml")/dblp/inproceedings[crossref=$p/@key])
    return <title>{fn:distinct-values($p/title/text())}</title>
} </answer>
```

6. Write a query that *maps* information for the year 2002 from **inproc.xml** and **proc.xml** into a schema that follows the pattern:

```
<publications>
  <author> *
    <name> .. </name>
    <in> *
      <forum> .. </forum>
      <article> .. </article>
    </in>
  </author>
</publications>
```

Again, you may wish to use fn:contains() or fn:ends-with().

```
<answer> <publications>{
    for $author in fn:distinct-values(doc("inproc.xml")/dblp/inproceedings/author
     [../year = "2002"])
    return <author>
             <name>{$author/text()}</name> {
             for $p in doc("inproc.xml")/dblp/inproceedings
              [author=$author and year = "2002"]
             return <in> {
                     for $c in doc("proc.xml")/dblp/proceedings
                      [@key=$p/crossref and year = "2002"]
                     return <forum>{$c/title/text()}</forum>
                     } <article>{$p/title/text()}</article>
                    </in>
            }</author>
}</publications></answer>
```

7. Sort papers in **inproc** in ascending alphabetical order of first author. When two papers were published by the same first author, further sort them in ascending alphabetical order of second author, etc. Results should be in the following format:

```
<answer>
    <paper>
      sorted papers here
    </paper>
</answer>
```

Since the Chamberlin XQuery tutorial does not cover the "order by" clause of XQuery (instead it uses the older sortby), see http://www.w3schools.com/xquery/xquery_select.asp for an example of sorting.

```
<answer>{
   for $p in doc("inproc.xml")/dblp/inproceedings
   order by $p/author[1], $p/author[2], $p/author[3], $p/author[4],
   $p/author[5], $p/author[6], $p/author[7], $p/author[8],
   $p/author[9], $p/author[10]
   return <paper>{$p}</paper>
}</answer>
```

Problem 2 [30 points]: Design XML schema for PBAY system.

Sellers(*sellerID*: int, *rating*: char, *email*: string)
Items(*itemID*: int, *type: string*)
Buyers(*buyerID*: int, *email*: string, *city*: string, *state*: string)
Stock(*itemID*: int, *sellerID*: int, *startBid*: float, *quantity*: int, *endingTime*: int)
Purchases(*itemID*: int, *buyerID*: int, *sellerID*: int, *price*: float, *purchaseQuantity*: int, *bidTime*: int)

The entity sets are **Sellers**, **Items** and **Buyers**. Make up some appropriate attributes for each entity set. Each of the entity sets should have 7 attributes. Specify keys and keyrefs. Explain how to encode **Stock** and **Purchases** relationship sets in XML schema.

```
Here's the basic schemas for Sellers, Items and Buyers. Your
answer need to specify keys and keyrefs clearly.

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

 <xsd:element name="Sellers">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="rating" type="xsd:char"/>
         <xsd:element name="email" type="xsd:string"/>
      </xsd:sequence>
      <xsd:key name="sellerID">
         <xsd:selector xpath=".//Sellers"/>
         <xsd:field xpath="./email"/>
      </xsd:key>
      <xsd:keyref name="itemRef" type="itemID">
        <xsd:selector xpath=".//Items"/>
      </xsd:keyref>
   </xsd:complexType>
 </xsd:element>

 <xsd:element name="Items">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="type" type="xsd:string"/>
      </xsd:sequence>
      <xsd:key name="itemID">
         <xsd:selector xpath=".//Items"/>
      </xsd:key>
   </xsd:complexType>
 </xsd:element>
```

```
  <xsd:element name="Buyers">
    <xsd:complexType>
       <xsd:sequence>
          <xsd:element name="email" type="xsd:string"/>
          <xsd:element name="city" type="xsd:string"/>
          <xsd:element name="state" type="xsd:string"/>
       </xsd:sequence>
       <xsd:key name="buyerID">
          <xsd:selector xpath=".//Buyers"/>
       </xsd:key>
       <xsd:keyref name="itemRef" type="itemID">
          <xsd:selector xpath=".//Items"/>
       </xsd:keyref>
       <xsd:keyref name="sellerRef" type="sellerID">
          <xsd:selector xpath=".//Sellers"/>
       </xsd:keyref>
    </xsd:complexType>
  </xsd:element>


</xsd:schema>
```