Introduction to Databases (summer term 2007)

Assignment 11

hand in on July 7, 2007 during the tutorial

This (a bit extended) assignment is intended only for those students who are interested in getting 8 ETCS points (instead of the normal 6) for the course Introduction to Databases. Besides answering the half of the below tasks correctly, of course the other criteria (30 % of the other tutorial assignment points and passing the exam) have to be fulfilled.

Besides the information on XML given in the lecture you may find some help in the additional XML resources linked from our homepage.

Task 11.1 (Some simple questions on XML) (2 Points)

a) For each XML document one can calculate the corresponding XML Information Set. How does this correspondence help in comparing two XML documents?

b) Consider a relational database (containing multiple tables with multiple rows).

Sketch, how this data could be represented using XML. Consider both mapping possibilities (data inside nested elements or in attributes).

Task 11.2 (XPath) (2 Points)

Consider a single XML document that stores information about books in a library. An example of such a file that shows the structure we assume is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bib>
  <book isbn="3-8274-1488-1" price="100">
    <name>Databases</name>
    <author>George Smith</author>
    <author>Joe Black</author>
    <chapter>
      <title>Introduction</title>
      <NoPages>16</NoPages>
    </chapter>
    <chapter>
      <title>Basics of Query Languages</title>
      <NoPages>50</NoPages>
    </chapter>
    <chapter>
      <title>The SQL Standard</title>
      <NoPages>40</NoPages>
    </chapter>
    <editor>Mike Blue</editor>
  </book>
  <book isbn="3-8274-3213-1" price="80">
    <name>Watching TV</name>
    <author>Freddy TV Junkie</author>
    <chapter>
      <title>Introduction</title>
      <NoPages>30</NoPages>
    </chapter>
    <chapter>
      <title>Switching On TV</title>
      <NoPages>50</NoPages>
    </chapter>
    <chapter>
      <title>Switching Off TV</title>
      <NoPages>2</NoPages>
    </chapter>
  </book>
</bib>
```

Introduction to Databases (summer term 2007)
Formulate suitable (absolute) XPath expressions that return a sequence of nodes of the XML document satisfying the following information demands:

a) All books that cost below 50€.
b) All authors who wrote a book with more than 5 chapters
c) All authors and editors of books with a chapter that is longer than 50 pages.
d) All authors who wrote a book that has chapter with the title ‘Evaluation’ between the chapters ‘Introduction’ and ‘Conclusion’.

Task 11.3 (XML Schema and XQuery)  
Consider the XML Schema document Lectures.xsd linked from our homepage which constraints the format of XML documents that describe a collection of lectures. These lectures can be part of several courses and can end with an oral or written examination.

a) Decide for the two sample XML documents linked from our homepage (Lectures1.xml and Lectures2.xml), whether they are valid according to the given schema. For the invalid document(s), give all reasons why they violate the schema constraints.

b) Describe the following query in XQuery:
List all lectures with an oral examination grouped by their instructors in the following format:

```
<lectures>
  <instructor name=INSTRUCTOR_NAME email=INSTRUCTOR_EMAIL>
    <lecture lang=LECTURE_LANG>LECTURE_TITLE</lecture>
  </instructor>
</lectures>
```

The instructors should be sorted in alphabetic order. Formulate your query against an XML document ‘lectures.xml’ that is considered to be valid according to the given schema.

Task 11.4 (XQuery)  
Although Oracle’s support for XML features can been as quite complete since the version 10g (XML schema, XML Query and XSL Transformations can be used), the handling of XML sometimes still is a bit cumbersome and slow. We therefore encourage you to develop the following XML Queries with the help of a public web frontend of a well-known LGPL licensed native XML server called eXist. This frontend can be accessed via the URL http://demo.exist-db.org/sandbox/sandbox.xql. As a data basis for the queries we employ the Mondial database, that can be inspected at http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-europe.xml. It stores a lot of information about countries, cities, lakes, etc. This data basis is accessible from the eXist front end under the path /mondial. To view the stored information about Germany we could e.g. use the XPath expression /mondial/country[name='Germany']. Please also take a look at the instructive saved XML query examples available in the combo box of the XQuery Sandbox frontend (starting with the 7th entry).

Formulate the following information demands as XML queries:

a) List the name and the inflation of all countries that have common border with Germany in the following format:

```
<GermanyNeighbours>
  <country>
    <name>...</name>
    <inflation>...</inflation>
  </country>
</GermanyNeighbours>
```
b) List all countries and their population where the percentage of Muslims is above 50 in the following format:

```xml
<MuslimCountries>
    <country name = "...">
        <population>...</population>
    </country>
</MuslimCountries>
```

c) List all cities (do not forget cities inside province elements) sorted by their population in descending order. Use the following output format:

```xml
<cities>
    <city>
        <name>...</name>
        <population>...</population>
    </city>
</cities>
```

Please note that in the current XML Query version that is implemented by eXist, the `sortBy` directive is replaced by `order by`. Since eXist does not support XML Schema yet, it treats numerical values as text which affects the sorting. Use the expression `order by xs:integer(Expr) descending` to cast Expr to an integer and sort the current node set according to it in descending order.

d) List all islands group by the country to which they belong. The sequence of countries should be sorted in descending order by the area the islands cover in this country. Use the following output format:

```xml
<countryInfo>
    <country name="..." islandArea="...">
        <island name="..." area="...">
    </island>
</country>
```

Task 11.5 (Structural Recursion with XSLT) (2 Points)
Linked from our homepage you find a small sample XML document ShopData.xml that contains some shop information about DVDs and Books.
Write an XSLT stylesheet that transforms an XML document of this kind to a simple price list of DVDs in the following format:

```xml
<DVDPrices>
    <DVD>
        <Title>...</Title>
        <Price>...</Price>
    </DVD>
</DVDPrices>
```

To develop and test your stylesheet you can e.g. use an evaluation version of Altova's well-known XML Suite XMLSpy1.

1http://www.altova.com/download/xmlspy/xml_editor_professional.html

Introduction to Databases (summer term 2007)
Task 11.6 (XQuery and XSLT)
Consider the following XML schema:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="book">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="title" type="xs:string"/>
                <xs:element name="author" type="xs:string" maxOccurs="3"/>
                <xs:element name="publicationYear" type="xs:year"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="journal">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="editor" type="xs:string" maxOccurs="5"/>
                <xs:element name="firstIssue" type="xs:string"/>
                <xs:element name="lastIssue" type="xs:string" minOccurs="0"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1. Formulate an XQuery against a valid document (let’s call it library.xml) of this type that lists for each author the books he/she has written and were published after 2004 and the number of journals he/she is an editor of in the following output format:

```xml
<result>
  <author name="...">
    <book title="..." publicationYear="...">
    <book title="..." publicationYear="...">
      ...
    </book>
  </author>
  <author name="..."> ...
  </author>
  ...
</result>
```
2. Consider the following two XSLT scripts:

a)  
```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="/library">
  <booklist>
    <xsl:apply-templates/>
  </booklist>
</xsl:template>

<xsl:template match="/library/book">
  <book>
    <xsl:apply-templates/>
  </book>
</xsl:template>

<xsl:template match="title">
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="publicationYear">
  <published>
    <xsl:value-of select="."/>
  </published>
</xsl:template>
</xsl:stylesheet>
```

b)  
```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="/library">
  <booklist>
    <xsl:apply-templates select="book"/>
  </booklist>
</xsl:template>

<xsl:template match="book">
  <book>
    <xsl:attribute name="title" select="title"/>
    <xsl:attribute name="publicationYear" select="publicationYear"/>
  </book>
</xsl:template>

<xsl:template match="publicationYear">
  <published>
    <xsl:value-of select="."/>
  </published>
</xsl:template>
</xsl:stylesheet>
```

They are intended to construct an XML document containing all books with their title and publication year in the following format:

```xml
<booklist>
  <book>
    <title>...</title>
    <published>...</published>
  </book>
</booklist>
```

Decide for each XSLT script whether it fulfills these requirements. Identify and correct all mistakes you find.