CSE 460 Project3: XML and XQuery

Due 23:59 12/05/2017 EST

Date 11/09/2017

1 Submission

Failure to comply the submission specifications will incur penalties for EACH violations

1.1 What to submit

A zip file has to be submitted through the 'submit_cse460' submission script by 12/05/2017 11:59PM EST. **ONLY** zip extension will be accepted.

1.2 Zip file naming convention

Write your *ubit* (**NO SPACE!**) for the filename, e.g. *jsmith.zip*, where *jsmith* is the ubit. The project is an **INDIVIDUAL** project, copied solutions will be considered violations of academic integrity.

1.3 Sub-structure of zip file

- On unzipping the zip file, there should be a folder named with your ubit *ubit*.
- Under the folder *ubit*, there should be one .xml file named prob1.xml, this file contains your answers for Problem 1, and one .txt file named prob2.txt, this file contains your answers for Problem 2.

2 Problem 1 (50 points) Database Design

You are asked to design an XML database for TinyFICO, which is a company focuses on credit scoring services. TinyFICO collects the bank account data of U.S. citizens and thus calculate the credit scores for the citizens.

TinyFICO has three major types of data to manage: customer data, bank information data and bank account data. More specifically, you are given below list of requirements:

- TinyFICO creates a record for each **customer**, and each record consists of:
  - An unique SSN, SSN is also the key for customer record.
  - A legal name.
  - Zero or more former name(s).
  - At least one address, each address consists of street, city, state, and zipcode.
  - At least one phone number.
  - A list of bank accounts that belong to the customer. (The requirements of bank account data will be given soon).
- TinyFICO creates a record for each **bank**, and each record consists of:
  - An unique bank code, which is also a key.
  - A bank name.
- TinyFICO also creates a record for each **bank account**, and each record consists of:
  - An unique bank account number, this account number is also the key.
  - A bank code, bank code indicates which bank the account belongs to.
  - A type, which indicates if the bank is a checking account, credit account, or a saving account, note only these three types are valid types for a bank account.

Write a DTD to model the above requirements. Use the XML Validation tool at `https://www.xmlvalidation.com/` to validate your DTD with your own testing XML data. For this problem, you need to submit a .xml file containing your DTD and your testing data that is valid against your DTD (note: testing data shouldn't be empty, and your testing data should be able to show all the modeling components of your DTD).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE TinyFico [
 <!ELEMENT TinyFico (customer*, bank*, bankAccount*)>
 <!ELEMENT customer (formerName*, address+, phone+, bAccount*)>
 <!ATTLIST customer SSN ID #REQUIRED
 legalName CDATA #REQUIRED>
 <!ELEMENT formerName (#PCDATA)>
 <!ELEMENT address (street,city,state,zipcode)>
 <!ELEMENT phone (#PCDATA)>
 <!ELEMENT street (#PCDATA)>
 <!ELEMENT city (#PCDATA)>
```

```
 <!ELEMENT state (#PCDATA)>
 <!ELEMENT zipcode (#PCDATA)>
 <!ELEMENT bAccount EMPTY>
 <!ATTLIST bAccount accountNum IDREF #REQUIRED>
 <!ELEMENT bank EMPTY>
 <!ATTLIST bank code ID #REQUIRED
 name CDATA #REQUIRED>
 <!ELEMENT bankAccount EMPTY>
 <!ATTLIST bankAccount accountNumber ID #REQUIRED
 bankCode IDREF #REQUIRED
 type (checking|saving|credit) #REQUIRED>
]>
<TinyFico>
    <customer SSN="S12345678" legalName="John Smith">
        <address>
        <street> 123 Maple</street>
        <city> Buffalo </city>
        <state> New York </state>
        <zipcode> 14260 </zipcode>
        </address>
        <phone> 123-456-7890
        </phone>
        <bAccount accountNum="C23456"/>
    </customer>
    <bank code="B10000000" name="Bank of America"/>
    <bank code="C20000000" name="Chase"/>
    <bankAccount accountNumber="C23456" bankCode="C20000000" type="credit"/>
</TinyFico>
```

Problem 2 (50 points) XQuery

Given *books.xml*, write below queries in your prob2.txt file, use XQuery comments to separate your answers, e.g. (: answer for 2.1 :). Use eXistDB to test and verify your answers, and the file path of books.xml should be **"/db/books.xml"**.

2.1 Find all the books co-authored by John and Mary.

```
let $b:=doc("/db/books.xml")/books
let $j:=
(for $a in $b/book
 where $a/author="John"
 return $a)

let $m:=
(for $a in $b/book
 where $a/author="Mary"
 return $a)
```

```
for $jb in $j, $mb in $m
where $jb/@id/string()=$mb/@id/string()
return $jb
```

2.2 Find all the books authored by John but **not** by Mary.

```
let $b:=doc("/db/books.xml")/books
let $j:=
(for $a in $b/book
 where $a/author="John"
 return $a)
for $book in $j
where every $author in $book/author satisfies $author ne "Mary"
return $book
```

2.3 Find the author names and the total numbers of their 2000 publications for all the authors who have publications in 2000, the result should be sorted in lexicographical order by the author names.

```
let $b:=doc("/db/books.xml")/books
let $bks:= (for $book in $b/book
where $book/year="2000"
return $book
)
let $bk :=<Books>{$bks}</Books>

for $book in $bk/book
let $bok := $book/author
group by $bok
order by $bok
return element author {
  element authorName { $bok/text() },
  element numberOfPublications { count($book) }
}
```