

CSCC43 Introduction to Databases (Summer 2009)

Assignment 3 - XML

Due (Electronic Copy): ~~Sunday, July 12~~ Thur., July 16, at 11:59pm

~~Due (Hard Copy):~~ ~~Monday, July 13, at 1:20pm~~

(You do not need to submit hard copy for this assignment)

Total Marks: 100

Weight: 10% of your final grade

The last 48 hours before the deadline are considered a silent period.

Student 1

NAME (LAST, FIRST) : _____

STUDENT NUMBER : _____

Student 2

NAME (LAST, FIRST) : _____

STUDENT NUMBER : _____

UTSC login name: _____

Problem 1 (15 points)

Consider following relational database schema for products and customers of a retail store, and as well as purchases made by the customers.

- Product(code, brand, category, weight, price, limit)
- Customer(id, name, phone, address, since)
- Purchase(transaction, cid, pcode, quantity, date)

The weight of a product is usually recorded in kg, but we should allow to use other units (such as in lb) as well. The price of a product is always in Canadian dollar. So there is no need to keep the currency information. The attribute limit in the relation Product specifies the maximal number of products of that brand and category can be purchased by a customer; the default limit value is 0 which means there is no limit on the product.

All attributes in the relation Customer are mandatory, except for phone, which is either a home or work phone (but not both) if presented. The attribute since records the date when the customer becomes a member of the company, which should be a date earlier than or same as the customer's first purchase date. An address value must contain street name and number, city and province, and possibly other fields. A customer may purchase 0 or more products within a day. The attribute cid and pcode in the relation Purchase are foreign keys referring to Customer.id and Product.code respectively.

1.1 (10 points)

Define a DTD, [store.dtd](#), for XML documents that list all products and customers, and as well as the purchases made by the customers, according the above relational database schema. Try to include as many of the specified constraints (both within schema and using words) as possible in DTD. Notice that the database instance shown above contains a few violations of some constraints, which are not easy to be specified in the DTD (you will have the chance to identify them using XQuery later on). Your DTD should start with:

```
<!ELEMENT Store (Products, Customers, Purchases)>
<!ELEMENT Products (Product+)>
...
```

Sample Solution

[See file store.dtd](#)

1.2 (5 points)

Provide an valid XML (with respect to store.dtd), [store.xml](#), that stores all the sample data in store.ddl. Your XML document should look something like:

```

<?xml version="1.0"?>
<!DOCTYPE Store SYSTEM "store.dtd">
<Store>
  <Products>
    <Product code="P80122804">
      <brand>EKTORP</brand><category>Sofa</category>
      <weight><value>59.5</value><unit>kg</unit></weight>
      <price>499.00</price>
    </Product>
    ...
  </Products>
  <Customers>
    <Customer id="C20016236798">
      <name>Bennett</name>
      <phone><home>416-6236798</home></phone>
      <address>
        <street>30 Charles St.</street><city> Toronto</city><province> ON</province>
        <others/>
      </address>
      <since>2003-05-26</since>
    </Customer>
    ...
  </Customers>
  <Purchases>
    <Purchase pcode="P80120089" cid="C20016236798" transaction="T100001">
      <quantity>1</quantity>
      <date>2003-05-26</date>
    </Purchase>
    ...
  </Purchases>
</Store>

```

Sample Solution

[See file store.xml](#)

Problem 2 (85 points)

Express the following questions in XQuery expressions.

Q1 (10 points)

Find the customers (their names and phone numbers) who have purchased product P60045393. This query must produce an output in following format:

```
<result>
```

```
<Customer name="Bennett" phone="416-6236798" />
</result>
```

Sample Solution

See file q1.xq

Q2a (15 points)

Find products purchased by each customer; sort customers in a descending order of their total numbers of purchased products. Note that a product may be purchased more than once by a customer - each purchase of the product should count one towards the total number. Also note that your result should include an entry for every customer even if he/she has not purchased any product. This query must produce an output in following format:

```
<result>
  <Customer id="C20016236798" numProds="4">
    <Prod pcode="P80120089" />
    <Prod pcode="P60045393" />
    <Prod pcode="P40085709" />
    <Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20014910013" numProds="4">
    <Prod pcode="P50104477" />
    <Prod pcode="P00094433" />
    <Prod pcode="P00094433" />
    <Prod pcode="P00094433" />
  </Customer>
  <Customer id="C20019302983" numProds="3">
    <Prod pcode="P80120089" />
    <Prod pcode="P40105298" />
    <Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20011342729" numProds="2">
    <Prod pcode="P80120089" />
    <Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20014398702" numProds="1">
    <Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20014827299" numProds="0" />
</result>
```

Sample Solution

See file q2a.xq

Q2b (5 points)

Consider Q2a again. This time we want to find the total number of **distinct** products purchased by each customer. Modify your answer to Q2a to produce this this result. This query must produce an output in following format:

```
<result>
  <Customer id="C20016236798" numDistProds="4">
<Prod pcode="P80120089" />
    <Prod pcode="P60045393" />
    <Prod pcode="P40085709" />
    <Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20019302983" numDistProds="3">
<Prod pcode="P80120089" />
    <Prod pcode="P40105298" />
    <Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20014910013" numDistProds="2">
<Prod pcode="P50104477" />
    <Prod pcode="P00094433" />
    <Prod pcode="P00094433" />
    <Prod pcode="P00094433" />
  </Customer>
  <Customer id="C20011342729" numDistProds="2">
<Prod pcode="P80120089" />
<Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20014398702" numDistProds="1">
<Prod pcode="P50104477" />
  </Customer>
  <Customer id="C20014827299" numDistProds="0" />
</result>
```

Sample Solution

[See file q2b.xq](#)

Q3 (15 points)

Find the product with a total revenue more than 1000 after 2003-01-01, together with a list the customers who have purchased the product. The total revenue of a product is calculated by the unit price times the total quantity it has been purchased by some customer during a period of time. This query must produce an output in following format:

```
<result>
  <Product code="P80120089" amount="1197">
```

```

    <Customer>
      <name>Bennett</name>
    </Customer>
  <Customer>
    <name>Alicia</name>
  </Customer>
<Customer>
  <name>Charles</name>
</Customer>
</Product>
</result>

```

Sample Solution

See file q3.xq

Q4 (15 points)

Find all the customers who made some purchase before registered with the company. The registration date is recorded in the attribute `since`. You may use the function `xs:date("2003-07-20")` to convert a plain string "2003-07-20" into a date, which can then be compared with other dates using `<`, `=`, `>` etc. This query must produce an output in following format:

```

<result>
  <Purchase date="2003-06-06">
    <Customer id="C20011342729" since="2003-07-20" />
  </Purchase>
</result>

```

Sample Solution

See file q4.xq

Q5 (5 points)

Explain what q5.xq will return, and show the actual result of the query.

Sample Solution

This XQuery expression finds all the violations of the constraint: "the number of products (of a particular brand and category) purchased by a customer cannot exceed the maximal number specified by the limit of that product. The actual result of this query is stored in file q5-output.xml

Q6 (20 points)

Study the file [q6-output.xml](#). Write an XQuery expression which restructures your original [store.xml](#) into [q6-output.xml](#).

Sample Solution

See file [q6.xq](#)

Submission instructions

You can work in groups of 2 people, with only one submission per group. Your submission must be typed. Handwritten assignments will not be accepted. The electronic submission must contain the following files (with these exact names):

1. [group.txt](#) - text file, with information (first and last names, student numbers) about all members of the group, and your UTSC login account; you can find a template of this file on the assignment page on the course website.
2. [a3.<extension>](#) - your solution to this assignment
 - you can use one of the acceptable formats ([.ps](#), [.pdf](#) or [plain ASCII file](#)); we prefer Postscript or PDF submissions (you may lost points for clarity and readability reasons). Submissions in other format will not be marked.
 - It is very important that you include in this file the information from [group.txt](#) as well (suggestion: include it in the first page of your file).
3. Check the assignments section on the course website for using submit command.

Suggestions

1. For suggestions on XML editor, see [here](#).
2. You can use any text editor you like, or you are familiar with. If you want to edit your assignment in LaTeX, you can find some useful information on the course website. You can use the example file provided there as a reference. If you choose to copy some examples from there, please acknowledge this in your assignment.