

Web-Informationssysteme, WS 2009/10
Übungsblatt 8

Besprechung am Di 22.12.2009

Aufgabe 8-1 XML Schema, ReferenzenDie Datei `bank.xml` enthält ein paar einfache Informationen über die Aktivitäten einer Bank:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <bank>
4
5   <accounts>
6     <savings_accounts>
7       <savings_account id="a1" interest="0.02">
8         <balance>2500</balance>
9       </savings_account>
10      <savings_account id="a2" interest="0.03">
11        <balance>15075</balance>
12      </savings_account>
13    </savings_accounts>
14
15    <checking_accounts>
16      <checking_account id="a3">
17        <balance>4025</balance>
18      </checking_account>
19      <checking_account id="a4">
20        <balance>-325</balance>
21      </checking_account>
22    </checking_accounts>
23  </accounts>
24
25  <customers>
26    <customer id="c1">
27      <name>Ben Richerdson</name>
28      <address>Park Drive 2</address>
29    </customer>
30    <customer id="c2">
31      <name>Angel Steady</name>
32      <address>Lake Sight 15</address>
33    </customer>
34  </customers>
35
36  <customer_accounts>
37    <customer_account c_id="c1" ac_id="a1"/>
38    <customer_account c_id="c1" ac_id="a3"/>
39    <customer_account c_id="c2" ac_id="a2"/>
40    <customer_account c_id="c2" ac_id="a4"/>
41  </customer_accounts>
42
43 </bank>
```

Schreiben Sie eine DTD und ein XML Schema für dieses Dokument, die folgende Anforderungen erfüllen sollen:

1. Es gibt zwei Typen von Konten: Spar- und Girokonten (*savings* und *checking account*).
2. Die id eines Kontos ist eindeutig unter allen Konten.
3. Die id eines Kunden ist eindeutig unter allen Kunden.
4. Die Attribute *c_id* und *ac_id* von *customer_account* verweisen auf Kunden bzw. Konten.
5. Der Kontostand jedes Kontos muss mehr als -5000 betragen.

Hinweis I: Im DTD-Formalismus können nicht alle diese Anforderungen ausgedrückt werden. In XML Schema ist dies jedoch der Fall. Verwenden Sie Vererbung für Giro- und Sparkonten!

Hinweis II: Sie können die erstellte DTD sowie die Schema-Definition mit *xmllint* überprüfen. Alternativ können Sie die Schema-Definition mit einem Schema-Validator testen:

<http://tools.decisionsoft.com/schemaValidate.html>

Aufgabe 8-2 XML Schema, Referenzen

Training

Die Dateien *publications.xml* und *publications.xsd* enthalten ein XML-Dokument mit Daten über Veröffentlichungen und deren Autoren sowie das zugehörige Schema in der Notation von XML Schema. Verändern Sie das Schema so, dass folgende Anforderungen erfüllt werden:

- a) In diesem Schema haben *masterthesis* und *phdthesis* den gleichen Typ. Finden Sie heraus, wie viele Autoren das Schema jeweils zulässt. Wie viele lässt unsere Prüfungsordnung zu? Passen Sie das Schema an, falls nötig.
- b) Es soll auch möglich sein, Informationen darüber zu repräsentieren, wenn eine Veröffentlichung eine andere zitiert.

Hinweis: Unter <http://en.wikipedia.org/wiki/BibTeX> finden Sie mehr über die Repräsentation von Veröffentlichungen und die dabei auftretenden Probleme.

Aufgabe 8-3 XPath, Einleitung

Konstruieren Sie ein (sinnvolles) XML-Dokument, auf dem die folgenden XPath-Ausdrücke die angegebenen Ergebnisse liefern:

Ausdruck	Ergebnis
1 <code>count(/descendant::*)</code>	7
2 <code>count(/descendant::*/*attribute::*)</code>	6
3 <code>count(/descendant::argument)</code>	2
4 <code>count(/descendant::class)</code>	1
5 <code>count(/descendant::method)</code>	2
6 <code>count(/descendant::method/*attribute::name)</code>	2
7 <code>count(/descendant::description)</code>	1
8 <code>count(/descendant::return)</code>	1
9 <code>count(/descendant::text())</code>	1
10 <code>count(/descendant::method/*child::return)</code>	1
11 <code>count(/descendant::method/*child::argument)</code>	2
12 <code>count(/descendant::class/*child::*)</code>	3
13 <code>count(/descendant::description/*child::*)</code>	0
14 <code>/descendant::class/*child::*[1]/self::description</code>	{"This class handles input"}
15 <code>/descendant::*/*attribute::name</code>	{"input", "text", "number"}
16 <code>/descendant::*/*attribute::type</code>	{"string", "int", "string"}
17 <code>/descendant::return/*attribute::type</code>	{"int"}
18 <code>/descendant::argument/*attribute::type</code>	{"string", "string"}
19 <code>boolean(/descendant::class/*attribute::name)</code>	true
20 <code>boolean(/descendant::return/*preceding-sibling::argument)</code>	false

Hinweis: Bei mengenwertigen Ausdrücken spielt die Reihenfolge der Ergebnisse eine Rolle (hier jeweils *document order*).