Klausur "Semistrukturierte Daten und XML"
Sommersemester 2009
Prof. Dr. Wolfgang May
16. Juli 2009, 14-16 Uhr
Bearbeitungszeit: 90 Minuten

Vorname:

Nachname:

Matrikelnummer:

Bei der Klausur sind **keine Hilfsmittel** (Skripten, Taschenrechner etc.) erlaubt. Handies müssen ausgeschaltet sein. Papier wird gestellt. Benutzen Sie nur die **ausgeteilten**, zusammengehefteten **Blätter** für Ihre Antworten. Schreiben Sie mit blauem/schwarzem Kugelschreiber, Füller etc.; Bleistift ist nicht erlaubt.

Zum **Bestehen** der Klausur sind **45** Punkte hinreichend.

☐ meine Note soll mit Matrikelnummer so bald wie möglich auf der Vorlesungs-Webseite veröffentlicht werden.

☐ meine Note soll nicht veröffentlicht werden; ich erfahre sie dann vom Prüfungsamt (bzw. für nicht von einem Prüfungsamt verwaltete Teilnehmer: beim Abholen des Scheins).

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (XPath, XQuery)	18	12
Aufgabe 2 (Entwurf, DTD)	16	12
Aufgabe 3 (Datenintegration)	20	10
Aufgabe 4 (XSLT)	16	6
Aufgabe 5 (Entwurf)	20	10
Summe	90	50

Note:

Die Klausur behandelt die Übernahme einer Kfz-Versicherung "B" durch eine Kfz-Versicherung "A".

(Disclaimer: einige Formulierungen sind juristisch gesehen nicht so ganz exakt.)

In den Aufgaben wird nur die Kraftfahrzeug-Haftpflicht betrachtet (die HP des Besitzers bezahlt Schäden an fremden Autos etc./ wenn er einen Unfall verursacht hat) . Kfz sind bundesweit nach Modell, Motorleistung und Baujahr in *Typklassen* eingeteilt (die sich jährlich ändern können). Soweit benutzen alle Versicherer dieselben Daten. Für die Klausur wird angenommen, dass diese in einer XML-Datei im Internet unter `http://kfz-hp.de/typklassen.xml` vorliegen; diese Datei ist auf dem letzten Blatt der Klausur angegeben. Für die Klausur wird vereinfacht angenommen, dass die Zuordnung auf Basis von Typ ("VW Golf"), Baujahr ("1998") und Leistung ("115" (kW)) vorgenommen wird.

Für jedes Auto schließt man einen eigenen Versicherungsvertrag ab (wenn jemand mehrere Autos besitzt, können diese bei unterschiedlichen Versicherungen versichert sein).

Jedem Ort ist eine *Regionalklasse* zugeordnet (Klausur: Wohnort des Besitzers). Diese Einteilung ist ebenfalls für alle Versicherer dieselbe (Klausur: anhand der ersten beiden Stellen der Postleitzahl).

Jeder Versicherer ordnet jeder Typklasse einen Basistarif (z.B. für Typklasse 18 sind dies bei Versicherung A 600€) und jeder Regionalklasse einen Prozentsatz (z.B. PLZ 10XXX Berlin 120%) zu.

Zu jedem einzelnen Vertrag ist die Schadenfreiheitsklasse (SF-Klasse) gespeichert (z.B. 5, entspricht grob der Anzahl Jahre wie lange kein Schadensfall von der Versicherung bezahlt werden musste). Aus der SF-Klasse ergibt sich, wieviel % des sich aus Basistarif und Regionalklasse ergebenden Betrages (z.B. 75%) die jeweilige Person pro Jahr bezahlen muss (siehe Bsp. weiter unten).

Versicherung A verwendet deutsche Begriffe für Element- und Attributnamen und speichert ihre Daten wie in dem zweiten XML-Beispieldokument auf der letzten Seite der Klausur angegeben.

Berechnungsbeispiel. Seppl Mosers Vertrag für seinen VW Golf, Bj. 1998, 115kW (Typklasse 18 → 600.00€ Basistarif), in 97999 Kuhdorf zugelassen (Regionalklasse 97 → 105% → 630.00€). Der Vertrag befindet sich in Schadenfreiheitsklasse 15 (d.h., Herr Moser hatte diesen Vertrag schon für ein vorheriges Fahrzeug) → 30% von 630€ → 189.00€.
Die Formel lautet also

$$\mathsf{Basisbetrag}(\mathsf{Typkl}(\mathit{fahrz.typ})) \cdot \mathsf{Prozent}(\mathsf{Regionalkl}(\mathit{wohnort}))\ /\ 100 \cdot \mathsf{Prozent}(\mathsf{Schadenfreiheitskl}(\mathit{vertrag}))\ /\ 100$$

Der Datenbestand von **Versicherung B** wird in Aufgabe 2 näher beschrieben und untersucht.

Aufgabe 1 (XPath, XQuery [18 Punkte])
Diese Aufgabe bezieht sich nur auf die Typklassen und Versicherung A (die XML-Dateien sind auf dem letzten Blatt gegeben). Geben Sie zu jedem Aufgabenteil eine XPath- oder XQuery-Anfrage an, die das Ergebnis in dem jeweils angegebenen Format ausgibt (falls kein Format angegeben, beliebig).

a) A-db: Geben Sie einen XPath- oder XQuery-Ausdruck an, der die Namen aller Kunden ausgibt, die einen Vertrag in SF-Klasse 20 oder höher haben. (2 P)

b) A-db: Geben Sie die Namen aller Kunden aus, die mindestens zwei Verträge haben und alle ihre Verträge in SF-Klasse 10 oder höher eingestuft sind. (4 P)

c) Geben Sie einen XQuery-Ausdruck an, der in Form von Tupeln (Kundenname, Kennzeichen, Betrag) für alle Verträge auflistet, wie hoch der zu bezahlende Versicherungsbetrag für den jeweiligen Vertrag (im aktuellen Jahr) ist, z.B.

<result name="Seppl Moser" kennz="KU-H 42" betrag="189.00" />

(Zugriff in XPath auf die Dokumente mit doc(*filename*)) (12 P)

Hinweis: Die ersten beiden Stellen der Postleitzahl bekommt man mit fn:substring(*plz*, 1, 2).

Lösung

a)
```
/A-db/vertrag[number(@inSFKlasse) > 19]/id(@kunde)/@name/string()

/A-db/kunde[@id = //vertrag[number(@inSFKlasse) > 19]/@kunde]
       /@name/string()
```

Hinweis: `distinct-value(...)` ist hier nicht notwendig, da die Auswertung von XPath in jedem Schritt die *Menge* der Knoten betrachtet, und dabei Duplikate eliminiert.

b)
```
for $k in //kunde
where (count(//vertrag[@kunde = $k/@id]) > 1)
  and (every $v in //vertrag[@kunde = $k/@id]
       satisfies $v/@inSFKlasse > 9)
return $k/@name/string()

for $k in //kunde
where (count(//vertrag[@kunde = $k/@id]) > 1)
  and not (//vertrag[@kunde = $k/@id and @inSFKlasse <= 9])
return $k/@name/string()

for $k in //kunde
let $v := //vertrag[id(@kunde) is $k]
where count($v) > 1 and not ($v[@inSFKlasse <= 9])
return $k/@name/string()
```

Hinweis (vgl. `//vertrag[id(@kunde) is $k]`) "=" testet auf String-Gleichheit (zwei Elemente ohne Content wären dabei immer gleich); "is" testet auf Knoten-Identitat.

c)
```
for $v in /A-db/vertrag
let $kunde := $v/id(@kunde),
    $typklasse := doc("typklassen.xml")//data
      [@typ=$v/auto/@typ and @jahr=$v/auto/@baujahr and @kw=$v/auto/@kw]
      /@typkl/string(),
    $typbetrag :=
       /A-db/typklasse[@nr=$typklasse]/number(@euro),
    $zip := $kunde/@plz,
    $regklasse := fn:substring($zip,1,2),
    $regprozent :=
       /A-db/regklasse[@plz=$regklasse]/number(@prozent),
    $sfklasse := $v/@inSFKlasse,
    $sfprozent :=
       /A-db/sfklasse[@nr=$sfklasse]/number(@prozent),
    $betrag := $typbetrag div 100 * $regprozent div 100 * $sfprozent
 return
 <result name="{$kunde/@name}" kennz="{$v/auto/@kennz}"
         betrag="{$betrag}"/>
```

```
for $v in /A-db/vertrag
let $typbetrag :=
       /A-db/typklasse
      [@nr=doc("typklassen.xml")//data
           [@typ=$v/auto/@typ and @jahr=$v/auto/@baujahr
               and @kw=$v/auto/@kw]
           /@typkl]/number(@euro),
    $regprozent :=
       /A-db/regklasse[@plz=fn:substring($v/id(@kunde)/@plz,1,2)]
         /number(@prozent),
    $sfprozent :=
       /A-db/sfklasse[@nr=$v/@inSFKlasse]/number(@prozent),
    $betrag := $typbetrag div 100 * $regprozent div 100 * $sfprozent
 return
 <result name="{$v/id(@kunde)/@name}" kennz="{$v/auto/@kennz}"
         betrag="{$betrag}"/>
```

```
for $v in /A-db/vertrag
return
 <result name="{$v/id(@kunde)/@name}" kennz="{$v/auto/@kennz}"
     betrag="{
       /A-db/typklasse
        [@nr=doc("typklassen.xml")//data
           [@typ=$v/auto/@typ and @jahr=$v/auto/@baujahr
               and @kw=$v/auto/@kw]
           /@typkl]/@euro
       div 100 *
       /A-db/regklasse[@plz=fn:substring($v/id(@kunde)/@plz,1,2)]/@prozent
       div 100 *
       /A-db/sfklasse[@nr=$v/@inSFKlasse]/@prozent
     }"/>
```

Die Verwendung von `number(...)` ist hier nicht notwendig (vgl. 3. Lösungsbeispiel), da XQuery an den arithmetischen Operatoren und an der Verwendung der Konstanten

100 erkennt, dass es sich um numerische Werte handelt.

`number(...)` benötigt man im wesentlichen dann, wenn man zwei Variablen vergleichen will, die an numerische Werte gebunden sind (vgl. Bsp. in Vorlesung) da XQuery das dann nicht statisch aus der Syntax der Anfrage ableiten kann.

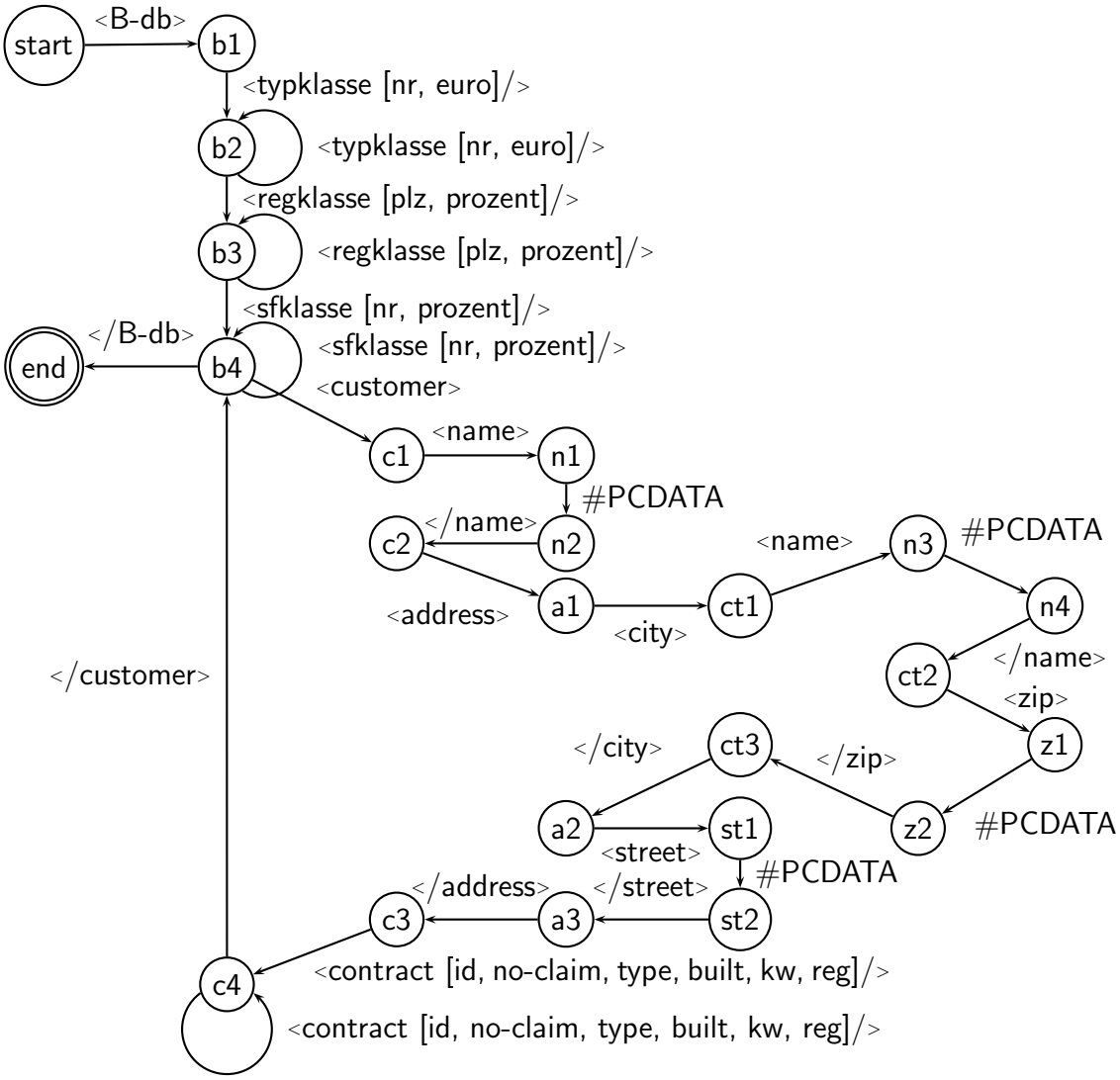
Umgekehrt wäre allerdings in 1a) vertrag[@inSFKlasse > "19"] *falsch*, da dies einen String-Vergleich mit dem String "19" verlangen würde, und alphanumerisch z.B. "8" > "19" ist!

Aufgabe 2 (Entwurf, DTD [16 Punkte])

Die Kundendatenbank von Versicherung B enthält prinzipiell dieselben Informationen wie bei Versicherung A. Für die Zuordnung von Fahrzeugtypen zu Typklassen wird ebenfalls die allgemein gültige Datei `http://kfz-hp.de/typklassen.xml` verwendet.

Die Struktur der Kundendatenbank von Versicherung B wird von dem unten angegebenen Automaten akzeptiert. Die Zuordnungen der Typ-, Regional- und SF-Klassen zu Basisbeträgen, Regional-% und SF-% (die im allgemeinen unterschiedlich zu denen von Versicherung A sind) sind bei B in derselben Form wie bei A (siehe `A-db.xml`) abgelegt. Damit unterscheidet sich effektiv nur die Struktur der kundenbezogenen Daten.

Für die kundenbezogenen Daten benutzt B englische Begriffe (zip = Postleitzahl, built ∼ Baujahr, no-claim ∼ SF-Klasse, registration ∼ Kennzeichen). Die Attribute sind in dem Automaten jeweils im öffnenden Tag angegeben.



1. Geben Sie eine möglichst feine, dem Automaten entsprechende DTD an. (8 P)

2. Geben Sie eine XML-Instanz an, die das Dokument illustriert (8 P) (Zuordnung von Basistarifen, Regional-Prozenten, SF-Prozenten [wobei die Prozentwerte im allgemeinen unterschiedlich von denen von Versicherung A sind], Kunden, Verträge), darunter

   - Karl Napf (derselbe, der seinen Mercedes 200D bei A versichert hat), hat einen Mercedes A150, Bj. 2005, 70 kW, Kennzeichen "EMD-A 150" in SF-Klasse 4 bei B versichert,
   - Michael Mustermann, wohnhaft am Rathausplatz 3, 80000 München, hat bei B zwei Fahrzeuge versichert:
     – einen BMW 320i, Bj. 2006, 110kW, Kennzeichen "M-BM 320", SF-Klasse 12, und
     – einen Golf IV, Bj. 2003, 85kW, Kennzeichen "M-M 0815" in SF-Klasse 0.

Lösung

```
<!ELEMENT B-db (typklasse+, regklasse+, sfklasse+, customer*)>
<!ELEMENT typklasse EMPTY>
  <!ATTLIST typklasse nr CDATA #REQUIRED
                      euro CDATA #REQUIRED>
<!ELEMENT regklasse EMPTY>
  <!ATTLIST regklasse plz CDATA #REQUIRED
                      prozent CDATA #REQUIRED>
<!ELEMENT sfklasse EMPTY>
  <!ATTLIST sfklasse nr CDATA #REQUIRED
                      prozent CDATA #REQUIRED>
<!ELEMENT customer (name, address, contract+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (city, street)>
<!ELEMENT city (name, zip)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT contract EMPTY>
  <!ATTLIST contract id ID #REQUIRED
                     no-claim CDATA #REQUIRED
                     type CDATA #REQUIRED
                     built CDATA #REQUIRED
                     kw CDATA #REQUIRED
                     reg CDATA #REQUIRED>
```

Es ist keine brauchbare Lösung, `typklasse/@nr` etc. als ID zu definieren, da diese Zahlen, z.B. "10" sowohl als Nummern von Typklassen als auch als PLZs von Regionalklassen und Nummern von SF-Klassen vorkommen. [Klausurbewertung: hier nur -1/2P; allerdings weitere -3P wenn man Aufgabe 1c) darauf basiert hat, weil damit die Lösung deutlich einfacher ist.]

Keines der Attribute in `contract` kann als IDREF ausgezeichnet werden (zum einen existiert z.B. `type` nicht als ID im selben Dokument, sondern nur in `typklassen.xml`, zum anderen ist dort auch nur (type, baujahr, kW) gemeinsam Schlüssel).

```
<?xml version="1.0"?>
<!DOCTYPE B-db SYSTEM "B-db.dtd">
<B-db>
 <typklasse nr="10" euro="290"/>
 <typklasse nr="11" euro="310"/>
     <!-- : -->
 <regklasse plz="10" prozent="90"/>
     <!-- : -->
 <regklasse plz="37" prozent="100"/>
     <!-- : -->
 <sfklasse nr="1" prozent="100"/>
     <!-- : -->
 <sfklasse nr="10" prozent="40"/>
     <!-- : -->
 <customer>
  <name>Michael Mustermann</name>
  <address><city><name>Muenchen</name>
                 <zip>80000</zip></city>
           <street>Rathausplatz 3</street></address>
  <contract id="B2006-389" no-claim="12" type="BMW 320i" built="2006"
      kw="110" reg="M-BM 320"/>
  <contract id="B2009-123" no-claim="0" type="VW Golf" built="2003"
      kw="85" reg="M-M 0815"/>
 </customer>
 <customer>
  <name>Karl Napf</name>
  <address><city><name>Fischtown</name>
                 <zip>21212</zip></city>
           <street>Deich 13</street></address>
  <contract id="B2005-765" no-claim="4" type="Mercedes A150" built="2005"
      kw="70" reg="EMD-A 150"/>
 </customer>
     <!-- : -->
</B-db>
```

Aufgabe 3 (Datenintegration [20 Punkte])
Diese Aufgabe baut auf Aufgabe 2 auf.

Versicherung A übernimmt Versicherung B. Damit werden alle Verträge übernommen. Versicherung A muss dazu die Kundendaten von Versicherung B in ihr XML-Dokument übernehmen. Die SF-Klassen der Verträge bleiben dabei unverändert. Für die Kunden gelten danach die Basistarife und Prozentbeträge wie für die Altkunden von A (wobei alle Kunden, für die dies teurer ist als vorher, ein Sonderkündigungsrecht haben).

a) Diese Frage dient der Vorüberlegung und ist nicht in XQuery zu beantworten, sondern einfach mit kurzem Text: Beschreiben Sie, welche Daten zu `A-db.xml` hinzugefügt werden müssen? (5 P)

b) Geben Sie eine XQuery-Anfrage oder ein XSLT-Stylesheet an, das alle drei Dokumente `typklassen.xml`, `A-db.xml` und `B-db.xml` verwendet (Zugriff in XPath: doc(*filename*)) und alle Elemente ausgibt, die zu `A-db.xml` hinzugefügt werden müssen. (die Elemente sollen noch dabei nur ausgegeben, nicht zu A-db hinzugefügt werden, da keine Sprache für XML-Updates behandelt wurde) (15 P)

Sie dürfen Funktionen local:getNewKundeId() und local:getNewVertragId(*Kennzeichen*) verwenden, die neue IDs erzeugen.

Hinweis: Nehmen Sie an, dass der Name eine Person eindeutig identifiziert.

Lösung

a) Für alle Kunden von B, die noch nicht bei A Kunde sind, müssen Elemente hinzugefügt werden. Alle Verträge von B müssen zu A hinzugefügt werden; dabei muss darauf geachtet werden, `vertrag/@kunde` jeweils auf die passende ID (Altkunden/Neukunden) zu setzen.

b)

```
(: die beiden Funktionen muessen nur da sein um es auszuprobieren :)
declare namespace fn = "http://www.w3.org/2005/xpath-functions";
declare function local:getNewKundeID($kunde as xs:string)
{ $kunde };
declare function local:getNewVertragID($kennz as xs:string)
{ fn:concat("vonB-",$kennz) };

let $newcustomers :=
  for $cust in doc("B-db.xml")//customer
  where not (doc("A-db.xml")//kunde[@name = $cust/name])
  return
    <kunde id="{local:getNewKundeID($cust/name)}"
           name="{$cust/name}"
           adresse="{$cust/address/street}"
           stadt="{$cust/address/city/name}"
           plz="{$cust/address/city/zip}" />
(: $newcustomers wird jetzt behalten :)
let $newcontracts :=
 (for $cont in doc("B-db.xml")//customer/contract
  let $cname := $cont/../name/text()
  (: kunde-referenzwert erzeugen :)
  let $reftokunde :=
        ( if ($cname = $newcustomers/@name)
          then $newcustomers[@name = $cname]/@id
          else doc("A-db.xml")//kunde[@name = $cname]/@id
        )
  return
    <vertrag id="{local:getNewVertragID($cont/@reg)}"
       kunde="{$reftokunde}" inSFKlasse="{$cont/@no-claim}">
      <auto typ="{$cont/@type}"
            baujahr="{$cont/@built}"
            kw="{$cont/@kw}"
            kennz="{$cont/@reg}"/>
    </vertrag> )
return
  ( $newcustomers,
    $newcontracts )
```

XSLT: prinzipiell genauso. `$newcustomers` wird dabei auch erst an eine Variable gebunden.

Aufgabe 4 (XSLT [16 Punkte])
Die Geschäftsleitung von A möchte sich einen Überblick über den Kundenbestand nach regionalen Gesichtspunkten geben lassen. Geben Sie ein XSLT-Stylesheet an, das z.B. mit

```
saxonXSL -s A-db.xml -xsl report.xsl
```

aufgerufen wird und eine HTML-Tabelle wie unten skizziert ausgibt:

- Gruppiert nach den beiden ersten PLZ-Stellen (aufsteigend)
- innerhalb der Gruppe alle in diesem Gebiet versicherten Fahrzeuge nach Modell, Baujahr, kW (Leistung), und SF-Klasse, aufsteigend nach Baujahr geordnet.

(Erzeugen Sie die richtige Struktur; Zentrierung, Linien etc. sind unwichtig. Wenn Sie sich bei bestimmten XSL-Kommandos nicht sicher sind, "erfinden" Sie eine *sinnvolle* Syntax.)

Region: *erste beide Stellen der PLZ*			
VW Golf	1998	115	15
Opel Astra	2000	80	13
:	:	:	:
Region: *erste beide Stellen der PLZ*			
:	:	:	:
:	:	:	:
	:		
	:		

Lösung

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fn="http://www.w3.org/2005/xpath-functions"
                version="2.0">
 <xsl:output method="html" indent="yes"/>

 <xsl:template match="A-db">
  <html>
   <body>
    <table>
     <xsl:apply-templates select="//regklasse">
      <!-- die sind zwar in A-db.xml schon aufsteigend geordnet,
            aber dies ist nirgends explizit garantiert.
            Sicher ist sicher. -->
      <xsl:sort select="@plz" data-type="number" order="ascending"/>
     </xsl:apply-templates>
    </table>
   </body>
  </html>
 </xsl:template>

 <xsl:template match="regklasse">
  <tr><td>Region:</td>
      <td><xsl:value-of select="@plz"/></td>
  </tr>
  <!-- alle Vertraege in der Region raussuchen -->
  <xsl:apply-templates
       select="//vertrag[fn:substring(id(@kunde)/@plz,1,2)
                          = current()/@plz]">
   <xsl:sort select="auto/@baujahr"
             data-type="number" order="ascending"/>
  </xsl:apply-templates>
 </xsl:template>

 <xsl:template match="vertrag">
  <tr> <td><xsl:value-of select="auto/@typ"/></td>
       <td><xsl:value-of select="auto/@baujahr"/></td>
       <td><xsl:value-of select="auto/@kw"/></td>
       <td><xsl:value-of select="@inSFKlasse"/></td>
  </tr>
 </xsl:template>
</xsl:stylesheet>
```

Hinweis: es gibt eine Alternativlösung wenn man nicht auf das "select" im "regklasse"-Template kommt: über alle Verträge laufen, nach Baujahr ordnen, und im "vertrag"-Template mit <xsl:if> testen, ob der Vertrag in der enteprechenden Regionalklasse liegt (die Regionalklasse muss man dann allerdings als Parameter mitgeben).

In der internen Auswertung ist dieses nicht unbedingt ineffizienter, da gute XSLT/XQuery-Engines Bedingungen ebenso nach "oben" ziehen, wie man es von relationalen Datenbanken kennt.

Aufgabe 5 (Entwurf [20 Punkte])
Ein Angestellter bei A schlägt vor, die Struktur der Datenbank zu ändern und "viel einfacher" zu machen:

```
<!ELEMENT Neu-db (vertrag*)>
<!ELEMENT vertrag (name, adresse, auto)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT adresse (#PCDATA)>
<!ELEMENT auto [wird unveraendert uebernommen]>
<!ATTLIST vertrag
          typBasisBetrag CDATA #REQUIRED
          regProzent CDATA #REQUIRED
          sfProzent CDATA #REQUIRED >
```

a) Geben Sie an, wie die Anfrage aus Aufgabe 1c) (Versicherungsbeitrag im aktuellen Jahr) zu diesem Entwurf aussehen würde. (5 P)

b) Beschreiben Sie, in welchen Punkten dieser Vorschlag nicht besonders gut ist. (8 P)

c) Viele Anwendungsprogramme basieren auf solch ungeeigneten Schemata bzw. DTDs. Nehmen Sie an, der obige Vorschlag ist vom Chef und Sie müssen ein Anwendungsprogramm für die Versicherung (z.B. in Java mit DOM) auf Basis dieser DTD realisieren. Wie lösen Sie das Problem durch Workarounds? Skizzieren Sie, was in Ihrem Entwurf das Anwendungsprogramm bei der Eintragung eines neuen Vertrages (z.B., wenn Hinz Kunz aus 53545 Linz einen VW Polo Bj. 2009 mit 75kW versichern lassen will) tun muss. (7 P)

Lösung

a)
```
for $v in /Neu-db/vertrag
let $betrag := $v/@typBasisBetrag *
               $v/@regProzent div 100 * $v/@sfProzent div 100
return
<result name="{$v/name}" kennz="{$v/auto/@kennz}" betrag="{$betrag}"/>
```

Die Verwendung von `number(...)` ist hier nicht notwendig, da XQuery an den arithmetischen Operatoren und an der Verwendung der Konstanten 100 erkennt, dass es sich um numerische Werte handelt.

b) Zum einen ist die Adresse jedes Kunden ggf. mehrfach gespeichert (das ist aber das geringere Problem).

Zum anderen fehlt die Zuordnung von Klassen zu Basistarifen bzw. Prozenten. Diese benötigt man wenn man einen neuen Vertrag eintragen will, um anhand von Fahrzeug, Wohnort und schadenfreien Jahren (im Falle eines Versicherungswechsels, sonst ja Neuvertrag und "0") die richtigen Werte einzutragen.

Die Basistarife und Regional-% könnte man prinzipiell aus den Einträgen (Typ [womit man mit `typklassen.xml` die Typklasse bekommt], Postleitzahl und Prozent) zurückrechnen (lineares Gleichungssystem lösen ...). Ist allerdings in einem bestimmten Typklasse oder in einer bestimmten Regionalklasse gerade kein laufender Vertrag vorhanden (bzw. wird der letzte gelöscht), sind die Daten verloren.

Bei Tarif-Updates müssen alle Verträge einzeln geupdated werden (in dem ursprünglichen Schema enthielten die Verträge keine konkreten Beträge und Prozentzahlen!).

Insider (ggf. Zusatzpunkte): Die SF-Klasse eines Vertrages kann man in der Realität nicht aus den Prozenten auf eine bestimmte Klasse zurückrechnen (z.B. haben SF20-24 25%, und man wird nur jedes Jahr eine Klasse hochgestuft, und kommt erst bei SF25 auf 20%). Hier fehlt wirklich Information! Mit diesem Schema kann man nicht mal korrekt die SF-Klassenzuordnung der Verträge (jedes Jahr eine SF-Klasse höher) verwalten!

c) Die Anwendung muss den Basistarif für jede Typklasse und die Prozente für Regional- und SF-Klassen separat irgendwo (dies kann z.B. ein separates XML-Dokument sein) ablegen. Bei einem Neueintrag müssen diese geholt und in dem neuen <contract>-Element abgelegt werden.

Das Problem der effektiv fehlenden Schadenfreiheitsklassen-Information kann man eigentlich nur lösen, indem man die DTD doch um ein Attribut zum Vertrag erweitert.

Trennen Sie diese Seite ggf. ab, um damit die restlichen Aufgaben bearbeiten zu können.

```
<!-- Dokument   http://kfz-hp.de/typklassen.xml -->
<typklassen>
 <data typ="VW Golf" jahr="2003" kw="85" typkl="13"/>
 <data typ="VW Golf" jahr="1998" kw="115" typkl="18"/>
 <data typ="VW Golf" jahr="2007" kw="147" typkl="23"/>
    <!-- : -->
 <data typ="Audi R8" jahr="2008" kw="386" typkl="25"/>
    <!-- : -->
 <data typ="Mercedes 200D" jahr="1981" kw="52" typkl="11"/>
 <data typ="Mercedes A150" jahr="2005" kw="70" typkl="14"/>
    <!-- : -->
</typklassen>
```

```
<?xml version="1.0"?>
<!DOCTYPE A-db SYSTEM "A-db.dtd">
<A-db>
 <typklasse nr="10" euro="300.00"/> <!-- Typklassen: Basisbetrag monoton steigend -->
 <typklasse nr="11" euro="315.00"/>
    <!-- : -->
 <typklasse nr="18" euro="600.00"/>
    <!-- : -->
 <typklasse nr="25" euro="1500.00"/>
    <!-- : -->                        <!-- Regionalklassen: -->
 <regklasse plz="10" prozent="120"/> <!-- Berlin -->
    <!-- : -->
 <regklasse plz="21" prozent="85"/>  <!-- irgendwo an der Kueste -->
    <!-- : -->
 <regklasse plz="97" prozent="105"/> <!-- irgendwo in Bayern -->
    <!-- : -->
 <sfklasse nr="0" prozent="100"/> <!-- SF-Klassen: Prozentwert monoton abnehmend -->
 <sfklasse nr="1" prozent="95"/>
    <!-- : -->
 <sfklasse nr="5" prozent="75"/>
    <!-- : -->
 <sfklasse nr="15" prozent="30"/>
    <!-- : -->
 <sfklasse nr="28" prozent="25"/>
    <!-- : -->
 <kunde id="A123" name="Seppl Moser"
   adresse="Lange Strasse 3" stadt="Kuhdorf" plz="97999"/>
 <kunde id="A456" name="Karl Napf"
   adresse="Deich 13" stadt="Fischtown" plz="21212"/>
    <!-- : -->
 <vertrag id="sm-95-497" kunde="A123" inSFKlasse="15">
   <auto typ="VW Golf" baujahr="1998" kw="115" kennz="KU-H 42"/>
 </vertrag>
 <vertrag id="sm-84-678" kunde="A123" inSFKlasse="5">
   <auto typ="Audi R8" baujahr="2008" kw="386" kennz="KU-R 8"/>
 </vertrag>
 <vertrag id="kn-81-007" kunde="A456" inSFKlasse="28">
   <auto typ="Mercedes 200D" baujahr="1981" kw="52" kennz="EMD-EN 123"/>
 </vertrag>
    <!-- : -->
</A-db>
```