

Klausur “Semistrukturierte Daten und XML”
Sommersemester 2008
Prof. Dr. Wolfgang May
16. Juli 2008, 10-12 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 (Entwurf, DTD)	23	18
Aufgabe 2 (XPath, XQuery)	22	14
Aufgabe 3 (XQuery oder XSLT)	15	9
Aufgabe 4 (XSLT)	20	12
Aufgabe 5 (Konzeptuelles)	10	0
Summe	90	53

Note:

Die Klausur behandelt die Datenbank einer Fluggesellschaft: Flugstrecken, Flugplan, Flugbuchungen.

Gegeben sind die folgenden Dinge:

- Ein Rahmen (siehe letzte Seite, wird in Aufgabe 1 vervollständigt)
- Daten zu Flughäfen. Da diese prinzipiell von allen Fluggesellschaften benutzt werden, wird angenommen, dass alle Flughafendaten weltweit (mit eindeutigen Flughafencodes) von der IATA (International Air Transport Assoc.) vorgegeben sind.

Neben den bereits eingefügten Flughäfen soll die Datenbasis die folgenden fluglinienspezifischen Informationen enthalten:

- die Flugstrecken der Gesellschaft: z.B. Flugnummer "LH123" von "FRA" nach "LIS", Abflug 12h30min, Ankunft 14h30min und Flugnummer "LH124" von "FRA" nach "JFK", Abflug 8h40min, Ankunft 11h20min (alle Zeiten geben die lokale Zeit an). Sommer- und Winterzeit werden nicht berücksichtigt.
- Annahme: Kein Flug geht über Mitternacht, d.h. alle Flüge landen an demselben Tag, an dem sie gestartet sind.
- Annahme: jede Flugstrecke wird täglich geflogen; hierzu sind also keine speziellen Daten notwendig.
- Für jeden Tag werden die Buchungen gespeichert. Buchungen werden, z.B. von Online-Web-Services etc., als XML-Fragmente in HTTP-Messages verschickt. Der Inhalt einer solchen Nachricht sieht folgendermaßen aus:

```
<booking flight="LH123" firstname="John" name="Doe" nationality="uk">  
  <date year="2008" month="07" day="16"/>  
</booking>
```

- Annahme: Name+Vorname identifizieren eine Person eindeutig.

Aufgabe 1 (Entwurf, DTD [23 Punkte])

Überlegen Sie sich, wie Sie Flüge und Buchungen in der XML-Datenbasis ablegen möchten. Überlegen Sie sich, wie Sie am besten den Datentyp "Zeit" (für Anflugs- und Ankunftszeit) in XML repräsentieren, damit Sie darauf auch (zumindest ein bisschen) rechnen können.

1. Erweitern Sie das angegebene XML-Fragment entsprechend Ihren Überlegungen mit Beispieldaten zu Flügen und Buchungen. (8 P)
2. Geben Sie die DTD zu Ihrem Entwurf an. (15 P)

Lösung Es gibt im wesentlichen zwei Varianten: bookings in jeweiligen flight geschachtelt (spart ein Join). Zeiten bildet man am besten entsprechend dem Beispiel für Datum bei Buchungen als

```
<departure hours="14" minutes="30"/>
```

ab.

Geschachtelte Variante:

```
<lufthansa>
  <airport ...> ... </airport>
  ;
  <flight nr="LH123" from="FRA" to="LIS">
    <departure hours="14" minutes="30"/>
    <arrival hours="14" minutes="30"/>
    <booking> ... </booking>
    <booking> ... </booking>
    :
  </flight>
  <flight nr="LH124" from="FRA" to="JFK">
    <departure hours="08" minutes="40"/>
    <arrival hours="11" minutes="20"/>
    <booking> ... </booking>
    :
  </flight>
  :
</lufthansa>
```

Flache Variante (mit ID/IDREF oder Key/Foreign Key-Semantik):

```
<?xml version="1.0"?>
<!DOCTYPE lufthansa SYSTEM "lufthansa.dtd">
<lufthansa>
  <airport code="FRA" city="Frankfurt/Main" country="D" timezone="+1">
    <name>Rhein-Main-Flughafen Frankfurt</name>
  </airport>
  <airport code="LON" city="London" country="GB" timezone="0">
    <name>London Heathrow</name>
  </airport>
  <airport code="LIS" city="Lisbon" country="P" timezone="0">
    <name>Lisboa Portela</name>
  </airport>
  <airport code="OP0" city="Porto" country="P" timezone="0">
    <name>Porto - Francisco Sa Carneiro</name>
  </airport>
  <airport code="JFK" city="New York" country="USA" timezone="-5">
    <name>John F. Kennedy Airport New York</name>
  </airport>
  <airport code="ALB" city="Albany" country="USA" timezone="-5">
    <name>Albany International Airport</name>
  </airport>
  :
  <flight nr="LH123" from="FRA" to="LIS">
    <departure hours="14" minutes="30"/>
    <arrival hours="14" minutes="40"/>
  </flight>
```

```

<flight nr="LH133" from="FRA" to="OP0">
  <departure hours="13" minutes="30"/>
  <arrival hours="14" minutes="20"/>
</flight>
<flight nr="LH136" from="MUC" to="OP0">
  <departure hours="15" minutes="30"/>
  <arrival hours="16" minutes="20"/>
</flight>
<flight nr="LH124" from="FRA" to="JFK">
  <departure hours="08" minutes="40"/>
  <arrival hours="11" minutes="20"/>
</flight>
<flight nr="LH125" from="JFK" to="ALB">
  <departure hours="14" minutes="10"/>
  <arrival hours="15" minutes="40"/>
</flight>
<flight nr="LH333" from="LIS" to="ALB">
  <departure hours="04" minutes="50"/>
  <arrival hours="07" minutes="40"/>
</flight>
:
<booking flight="LH123" firstname="John" name="Doe" nationality="uk">
  <date year="2008" month="07" day="16"/>
</booking>
<booking flight="LH123" firstname="John" name="Doe" nationality="uk">
  <date year="2008" month="07" day="23"/>
</booking>
<booking flight="LH124" firstname="Hans" name="Mueller" nationality="de">
  <date year="2008" month="07" day="16"/>
</booking>
<booking flight="LH124" firstname="Peter" name="Schmidt" nationality="de">
  <date year="2008" month="07" day="16"/>
</booking>
<booking flight="LH125" firstname="Hans" name="Mueller" nationality="de">
  <date year="2008" month="07" day="16"/>
</booking>
<booking flight="LH125" firstname="Peter" name="Schmidt" nationality="de">
  <date year="2008" month="07" day="17"/>
</booking>
</lufthansa>

```

DTD der geschachtelten Variante:

```
<!ELEMENT lufthansa (airport*, flight*, booking*)>
<!ELEMENT airport (name)>
  <!ATTLIST airport code ID #REQUIRED
                    city CDATA #REQUIRED
                    country NMTOKEN #REQUIRED
                    timezone CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT flight (departure, arrival, booking*)>
  <!ATTLIST flight nr ID #REQUIRED
                  from IDREF #REQUIRED
                  to IDREF #REQUIRED>
<!ELEMENT departure EMPTY>
  <!ATTLIST departure hours CDATA #REQUIRED
                    minutes CDATA #REQUIRED>
<!ELEMENT arrival EMPTY>
  <!ATTLIST arrival hours CDATA #REQUIRED
                    minutes CDATA #REQUIRED>
<!ELEMENT booking (date)>
  <!ATTLIST booking name CDATA #REQUIRED
                    flight IDREF #REQUIRED
                    firstname CDATA #REQUIRED
                    nationality NMTOKEN #REQUIRED>
<!ELEMENT date EMPTY>
  <!ATTLIST date year CDATA #REQUIRED
                month CDATA #REQUIRED
                day CDATA #REQUIRED>
```

Aufgabe 2 (XPath, XQuery [22 Punkte])

Bearbeiten Sie die folgenden Aufgabenteile (1)–(4) entsprechend der von Ihnen entworfenen Datenbasis. 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).

1. Geben Sie einen XPath-Ausdruck an, der die Flugnummern aller Lufthansa-Flüge ausgibt, die *John Doe* im Juli 2008 gebucht hat. (3 P)

Lösung Für die flache Version:

```
//flight[@nr = //booking[@firstname="John" and @name="Doe"
and date[@month="07" and @year="2008"]]/@flight/string(@nr)
```

oder

```
//booking[@firstname="John" and @name="Doe"
and date[@month="07" and @year="2008"]]/id(@flight)//string(@nr)
```

(diese beiden entfernen Duplikate automatisch (Knotenmenge der flights)), oder

```
distinct-values(//booking[@firstname="John" and @name="Doe"
and date[@month="07" and @year="2008"]]/string(@flight))
```

wobei man Duplikat-Werte explizit entfernen muss.

Für die hierarchische Version ist der Ausdruck einfacher:

```
//flight[booking[@firstname="John" and @name="Doe"
and date[@month="07" and @year="2008"]]]/string(@nr)
```

2. Geben Sie die Namen aller Flughäfen an, von denen man Albany (Code: ALB) mit genau einer Zwischenlandung erreichen kann. Die Ausgabe darf Duplikate enthalten (5 P).

Lösung Die Idee, eine navigative Lösung zu versuchen, ist naheliegend, aber nicht einfach. Sie enthält auch keine Duplikate:

```
//airport[@code =
//flight[@to = //flight[@to="ALB"]/@from]/@from]
/name/text()
```

Mit einem Join ist es einfacher, es enthält aber zuerst einmal noch Duplikate:

```
for $f1 in //flight,
    $f2 in //flight
where $f2/@to="ALB" and $f2/@from = $f1/@to
return $f1/id(@from)/name/text()
```

Oder so:

```
for $f2 in //flight[@to="ALB"],
    $f1 in //flight[@to=$f2/@from]
return $f1/id(@from)/name/text()
```

Duplikate kann man entfernen, wenn man alles noch in ein distinct-values einpackt:

```
distinct-values(
for $f2 in //flight[@to="ALB"],
    $f1 in //flight[@to=$f2/@from]
return $f1/id(@from)/name/text())
```

Anmerkung: das distinct-values darf nicht innerhalb des returns stehen - dort hilft es auch wieder nichts.

3. Geben Sie die Namen aller Flughäfen an, von denen man alle portugiesischen Flughäfen (Landescode: "P") direkt erreichen kann. (5 P)

Lösung

```
for $a in //airport
where every $port in //airport[@country="P"]
    satisfies //flight[id(@from) is $a and id(@to) is $port]
return $a/name
```

Wenn man mit every nicht zurechtkommt, kann man es hier auch als Menge behandeln:

```
let $c := count(//airport[@country="P"])
for $a in //airport
let $reachable := //flight[id(@from) is $a]/id(@to)[@country="P"]
where count($reachable) = $c
return $a/name
```

```
let $port := //airport[@country="P"]
for $a in //airport
let $reachable := //flight[id(@from) is $a]/id(@to)[@country="P"]
where deep-equal($port, $reachable)
return $a/name
```

Anmerkung: "=" vergleicht die String-Werte; "is" vergleicht auf Knotenidentität.

4. Welche Passagiere fliegen am 16.7.2008 aus einem anderen Land in die USA, und am nächsten Tag mit einem Inlandsflug weiter? Geben Sie die Namen der Passagiere sowie die Flugnummern an (9 P).

Lösung

```

for $b1 in
  //booking[date/@day="16" and date/@month="07" and date/@year="2008"
    and id(@flight)/id(@to)/@country="USA"
    and id(@flight)/id(@from)/@country!="USA"],
  $b2 in
  //booking[date/@day="17" and date/@month="07" and date/@year="2008"
    and id(@flight)/id(@from)/@country="USA"
    and id(@flight)/id(@to)/@country="USA"]
where $b1/@name = $b2/@name and
  $b1/@firstname = $b2/@firstname and
  $b1/id(@flight)/@to = $b2/id(@flight)/@from
return
  <result flight="{ $b1/@flight}" flight2="{ $b2/@flight}"
    name="{ $b1/@name}" firstname="{ $b1/@firstname}"/>

```

Man kann auch noch die where-Klausel in die Bedingung von \$b2 nehmen, und so ein *korreliertes* Join verwenden.

Aufgabe 3 (XQuery oder XSLT [15 Punkte])

Es soll eine HTML-Tabelle (Flugnummer, Start, Ziel) erstellt werden, die genau diejenigen Flüge enthält, die mindestens 8 Stunden dauern.

1. Diese Frage dient der Vorüberlegung und ist nicht in XQuery zu beantworten, sondern einfach mit einem Wert: Wieviel Uhr ist es in Frankfurt, wenn Flug LH123 in Lissabon landet? (2 P)
2. Geben Sie eine XQuery-Anfrage oder ein XSLT-Stylesheet an, das die gewünschte Tabelle erzeugt. (13 P)

Lösung XQuery:

Wenn man Stunden/Minuten in getrennten Attributen ablegt (was man bei strukturierten Datentypen immer tun sollte, wenn es keinen built-in Type gibt), kann man diese gut verwenden:

```
<html>
  <body>
    <table>
      { for $f in //flight
        let $t1 := number(id($f/@from)/@timezone),
            $t2 := number(id($f/@to)/@timezone),
            $timediff := $t1 - $t2,
            $durationHours :=
              if (number($f/departure/@minutes) < number($f/arrival/@minutes))
              then $f/arrival/@hours - $f/departure/@hours + $timediff
              else $f/arrival/@hours - $f/departure/@hours + $timediff - 1
        where $durationHours > 7
        return
          <tr><td>{string($f/@nr)}</td>
            <td>{string($f/@from)}</td>
            <td>{string($f/@to)}</td>
          </tr>
      }
    </table>
  </body>
</html>
```

Das "if" kann man auch als "or" in der where-Klausel unterbringen.

Man kann die Dauer auch komplett in Stunden oder Minuten berechnen, muss dann aber berücksichtigen, dass die Zeitzone in Stunden angegeben ist, und eine Stunde 60 Minuten hat.

Dies wird im folgenden XSLT-Stylesheet verwendet:

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="2.0">
  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="lufthansa">
    <html>
      <body>
        <table>
          <tr><th>Flugnummer</th><th>von</th><th>nach</th></tr>
          <xsl:apply-templates select="flight"/>
        </table>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="flight">
    <xsl:variable name="t1" select="number(id(@from)/@timezone)"/>
    <xsl:variable name="t2" select="number(id(@to)/@timezone)"/>
    <xsl:variable name="timediff" select="$t1 - $t2"/>
    <xsl:variable name="duration"
      select="arrival/@hours * 60 + arrival/@minutes
             - departure/@hours * 60 - departure/@minutes
             + $timediff * 60"/>
    <xsl:if test="$duration >= 480">
      <tr><td><xsl:value-of select="@nr"/></td>
        <td><xsl:value-of select="@from"/></td>
        <td><xsl:value-of select="@to"/></td>
      </tr>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>

```

Wenn man Zeitangaben als atomare Werte, z.B. abflug="1430" modelliert hat (was eindeutig die schlechtere Alternative ist), müsste man eigentlich erst string-basierte Extraktionsfunktionen definieren, um damit arbeiten zu können. Im vorliegenden Fall genügt aber sogar,

```

$duration := @ankunft - @abflug + $timediff * 100 und
$duration > 800

```

zu verwenden (obwohl $1120-840+6*100 = 880$ offensichtlich nicht 8h80min, und auch nicht 8.8h, sondern 8h:40 Minuten entspricht).

Aufgabe 4 (XSLT [20 Punkte])

Jede Fluggesellschaft muss für ihre Flüge, die auf einem US-amerikanischen Flughafen landen, zwei Tage vorher einen Report übermitteln, der diese Flüge mit den Namen aller Passagiere gemäß folgender DTD enthält:

```
<ELEMENT airlineReport (name, flight+)>
  <ATTLIST airlineReport month CDATA #REQUIRED
                        day CDATA #REQUIRED
                        year CDATA #REQUIRED>
<ELEMENT name (#PCDATA)> <!-- name of the airline that submits the report -->
<ELEMENT flight (person+)>
  <ATTLIST flight code ID #REQUIRED>
<ELEMENT person (givenname, familyname, nationality)>
  <ELEMENT givenname (#PCDATA)>
  <ELEMENT familyname (#PCDATA)>
  <ELEMENT nationality (#PCDATA)>
```

1. Geben Sie ein kurzes charakteristisches XML-Fragment an, wie dieser Report für die Lufthansa aussieht. (3 P)
2. Schreiben Sie ein XSLT-Stylesheet, das z.B. mit

```
saxonXSL -s lufthansa.xml -xsl makeUSreport.xsl month=07 day=16 year=2008
```

aufgerufen wird und für den angegebenen Tag den entsprechenden Report der Lufthansa entsprechend Ihrer Datenbasis erstellt. (17 P)

Lösung Der report sieht so aus:

```
<airlineReport month="07" day="16" year="2008">
  <name>Lufthansa</name>
  <flight code="LH123">
    <person>
      <givenname>John</givenname>
      <familyname>Doe</familyname>
      <nationality>uk</nationality>
    </person>
    :
  </flight>
  :
</airlineReport>
```

Für die flache (in diesem Fall etwas schwierigere) Variante:

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="2.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:param name="month"/>
  <xsl:param name="day"/>
  <xsl:param name="year"/>
  <xsl:template match="lufthansa">
    <airlineReport month="{ $month}" day="{ $day}" year="{ $year}">
      <name>Lufthansa</name>
      <xsl:apply-templates select="flight[id(@to)/@country='USA']"/>
    </airlineReport>
  </xsl:template>
  <xsl:template match="flight">
    <xsl:variable name="code" select="@nr"/>
    <flight code="{@nr}">
      <xsl:apply-templates select="//booking[@flight=$code and
        date/@day=$day and date/@month=$month and date/@year=$year]"/>
    </flight>
  </xsl:template>
  <xsl:template match="booking">
    <person>
      <givenname> <xsl:value-of select="@firstname"/> </givenname>
      <familyname> <xsl:value-of select="@name"/> </familyname>
      <nationality> <xsl:value-of select="@nationality"/> </nationality>
    </person>
  </xsl:template>
</xsl:stylesheet>

```

Anstelle der Variablen im flight-template kann man auch "current()" verwenden, um die zum Flug gehörigen Buchungen zu selektieren:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="2.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:param name="month"/>
  <xsl:param name="day"/>
  <xsl:param name="year"/>
  <xsl:template match="lufthansa">
    <airlineReport month="{ $month}" day="{ $day}" year="{ $year}">
      <name>Lufthansa</name>
      <xsl:apply-templates select="flight[id(@to)/@country='USA']"/>
    </airlineReport>
  </xsl:template>
  <xsl:template match="flight">
    <flight code="{@nr}">
      <xsl:apply-templates select="//booking[@flight=current()/@nr and
        date/@day=$day and date/@month=$month and date/@year=$year]"/>
    </flight>
  </xsl:template>
  <xsl:template match="booking">
    <person>
      <givenname> <xsl:value-of select="@firstname"/> </givenname>
      <familyname> <xsl:value-of select="@name"/> </familyname>
      <nationality> <xsl:value-of select="@nationality"/> </nationality>
    </person>
  </xsl:template>
</xsl:stylesheet>
```

Aufgabe 5 (Konzeptuelles [10 Punkte])

In einer deklarativen Datenbank-Sprache kann man nicht nur Anfragen stellen, sondern auch Daten ändern, löschen und einfügen. Letzteres geschieht z.B. in SQL durch das Konstrukt

```
UPDATE table_name
SET attribute = new_value
WHERE condition .
```

Welche Informationen muss eine solche Sprache (denken Sie z.B. an eine Ergänzung zu XQuery) zum Einfügen, Löschen, oder Ändern von Werten in XML-Dokumenten ausdrücken können?

Wie würden Sie in einer solchen Sprache eine Flugplanänderung, z.B. die Ankunftszeit von LH123 auf 14:40 Uhr zu setzen, ausdrücken?

Lösung Man muss angeben welche(r) Knoten geändert/gelöscht werden soll(en) (z.B. durch einen XPath-Ausdruck), welche Operation (delete, insert after/before, update) ausgeführt werden soll, und welches (bei update oder insert) der neue Wert ist.

Mögliche Anweisungskonstrukte könnten z.B. so aussehen:

```
update //flight[@nr="LH123"]/arrival/@minutes to "40"
```

```
update //flight
set arrival/@minutes to "40"
where @nr="LH123"
```

```
for $f in //flight
where $f/@nr="LH123"
set $f/arrival/@minutes to "40"
```

```
Loeschungen saehen dabei etwa so aus:
delete //flight[@nr="LH123"]
delete from //flight where @nr="LH123"
```

Wichtig ist hierbei, relativen Teilausdrücken (wie "@nr='LH123'" oder "arrival/@minutes") entweder als relative Ausdrücke stehenzulassen und zu (implizit) auf dem aktuell bearbeiteten Knoten zu beziehen, oder explizite Variablen zu benutzen.

Aus diesen Grund ist z.B.

```
FALSCH  update //flight
FALSCH  set //flight/arrival/@minutes to "40"
FALSCH  where //flight/@nr="LH123"
```

nicht praktikabel, da die Zuordnung verlorengeht.

Trennen Sie diese Seite ggf. ab, um damit die restlichen Aufgaben bearbeiten zu können (Kopien dieses Blattes sind verfügbar; versehen Sie diese bitte zuerst mit Ihrem Namen).

```
<lufthansa>
  <airport code="FRA" city="Frankfurt/Main" country="D" timezone="+1">
    <name>Rhein-Main-Flughafen Frankfurt</name>
  </airport>
  <airport code="LON" city="London" country="GB" timezone="0">
    <name>London Heathrow</name>
  </airport>
  <airport code="LIS" city="Lisbon" country="P" timezone="0">
    <name>Lisboa Portela</name>
  </airport>
  <airport code="JFK" city="New York" country="USA" timezone="-5">
    <name>John F. Kennedy Airport New York</name>
  </airport>
  <airport code="ALB" city="Albany" country="USA" timezone="-5">
    <name>Albany International Airport</name>
  </airport>
  <!-- weitere Flughäfen -->
```

```
</lufthansa>
```