

**Übung zur Vorlesung  
„Einsatz und Realisierung von Datenbanksystemen“  
im Sommersemester 2007**

Richard Kuntschke (richard.kuntschke@in.tum.de)

**Lösungen zu Blatt 11**

**Aufgabe 1**

Das XML-Schema und die Beispielausprägung für die *Universität der großen Denker* finden Sie auf der Webseite zur Übung unter:

<http://www-db.in.tum.de/teaching/ss07/impldb/exercises/>

Basierend auf dieser Beispielausprägung können die Anfragen in XQuery beispielsweise wie folgt formuliert werden:

1. Um die Professoren zu ermitteln, die Vorlesungen lesen, werden die ProfessorIn-Elemente der Ausprägung sukzessive betrachtet und getestet, ob sie Vorlesungen-Unterelemente aufweisen.

```
<ProfessorenMitLehrbelastung>
{ for $p in doc("Uni.xml")/Universität/Fakultäten/
  Fakultät/ProfessorIn[Vorlesungen]
  return
  <ProfessorIn PersNr="{ $p/@PersNr }">
    { $p/Name/text() }
  </ProfessorIn> }
</ProfessorenMitLehrbelastung>
```

Führt man diese Anfrage auf der Ausprägung aus, so erhält man folgendes Resultat:

```
<ProfessorenMitLehrbelastung>
  <ProfessorIn PersNr="P2134">Augustinus</ProfessorIn>
  <ProfessorIn PersNr="P2125">Sokrates</ProfessorIn>
  <ProfessorIn PersNr="P2126">Russel</ProfessorIn>
  <ProfessorIn PersNr="P2133">Popper</ProfessorIn>
  <ProfessorIn PersNr="P2137">Kant</ProfessorIn>
</ProfessorenMitLehrbelastung>
```

2. Diese Anfrage kann mittels einer Mengendifferenz relativ einfach formuliert werden. Dazu wird überprüft, ob für Studenten die Differenz der Menge aller Vorlesungen und der Menge der gehörten Vorlesungen leer ist. Eine Mengendifferenz kann in XQuery mittels der Funktion *except* ausgedrückt werden; die Leere-Menge-Eigenschaft mittels *empty*:

```

<hörenAlleVorlesungen>
{ for $s in doc("Uni.xml")/Universität/Studenten/Student
  where empty( doc("Uni.xml")/Universität//Vorlesung
    except id($s/hört/@Vorlesungen))
  return
    <Student MatrNr="{ $s/@MatrNr}">
      { $s/Name/text() }
    </Student> }
</hörenAlleVorlesungen>

```

Ausgeführt auf unserer Ausprägung ergibt sich ein leeres Element, da es keine Studenten gibt, die alle Vorlesungen hören:

```

<hörenAlleVorlesungen/>

```

3. Die Anfrage lässt sich analog zu Teilaufgabe 2 formulieren. Hinzu kommt einzig ein Knotentest auf vierstündige Vorlesungen, d.h. das Prädikat [./SWS=4]:

```

<hörenAlle4SWSVorlesungen>
{ for $s in doc("Uni.xml")/Universität/Studenten/Student
  where empty( doc("Uni.xml")/Universität//Vorlesung[./SWS=4]
    except id($s/hört/@Vorlesungen))
  return
    <Student MatrNr="{ $s/@MatrNr}">
      { $s/Name/text() }
    </Student> }
</hörenAlle4SWSVorlesungen>

```

Da es keine Studenten gibt, die alle vierstündigen Vorlesungen hören, ist das Ergebnis der Anfrage wiederum leer:

```

<hörenAlle4SWSVorlesungen/>

```

4. Ähnlich zur Anfrageformulierung in SQL kann in XQuery die Aggregatfunktion **max** verwendet werden. Die Aggregation wird über die Menge aller Studenten ( $\$m$  im let-Abschnitt) durchgeführt:

```

<EwigeStudenten>
{ for $s in doc("Uni.xml")/Universität/Studenten/Student
  let $m := doc("Uni.xml")/Universität/Studenten/Student
  where $s/Semester = max($m/Semester)
  return
    <Student MatrNr="{ $s/@MatrNr}" Semester="{ $s/Semester/text()}">
      { $s/Name/text() }
    </Student> }
</EwigeStudenten>

```

In unserer Universität gibt es einen Studenten der dieses Kriterium erfüllt:

```

<EwigeStudenten>
  <Student MatrNr="M24002" Semester="18">Xenokrates</Student>
</EwigeStudenten>

```

5. Die Berechnung der Gesamtzahl der Semesterwochenstunden entspricht einer Aggregation mittels der **sum** Funktion.

```

<ProfessorenLehrbelastung>
{ for $p in doc("Uni.xml")/Universität/Fakultäten/
  Fakultät/ProfessorIn
  let $v := $p/Vorlesungen/Vorlesung
  return
  <ProfessorIn PersNr="{ $p/@PersNr }">
    { $p/Name/text() }
    <Lehrbelastung>{sum($v/SWS)}</Lehrbelastung>
  </ProfessorIn> }
</ProfessorenLehrbelastung>

```

Da wir keinen Knotentest auf ProfessorIn durchführen, um zu überprüfen, ob es ein Kindelement Vorlesungen gibt, sind auch die Professoren im Ergebnis enthalten, die keine Vorlesungen halten.

```

<ProfessorenLehrbelastung>
  <ProfessorIn PersNr="P2134">Augustinus<Lehrbelastung>2</Lehrbelastung>
</ProfessorIn>
  <ProfessorIn PersNr="P2136">Curie<Lehrbelastung>0</Lehrbelastung>
</ProfessorIn>
  <ProfessorIn PersNr="P2127">Kopernikus<Lehrbelastung>0</Lehrbelastung>
</ProfessorIn>
  <ProfessorIn PersNr="P2125">Sokrates<Lehrbelastung>10</Lehrbelastung>
</ProfessorIn>
  <ProfessorIn PersNr="P2126">Russel<Lehrbelastung>6</Lehrbelastung>
</ProfessorIn>
  <ProfessorIn PersNr="P2133">Popper<Lehrbelastung>2</Lehrbelastung>
</ProfessorIn>
  <ProfessorIn PersNr="P2137">Kant<Lehrbelastung>10</Lehrbelastung>
</ProfessorIn>
</ProfessorenLehrbelastung>

```

6. Folgende XQuery-Anfrage setzt die Aufgabenstellung wortgetreu um:

```

<NichtSoGuteStudenten>
{ for $s in doc("Uni.xml")//Student
  [not(descendant::Prüfung[@Note < 3.0])]
  return
  <Student MatrNr="{ $s/@MatrNr }">
    { $s/Name/text() }
    </Student> }
</NichtSoGuteStudenten>

```

Allerdings werden dadurch auch die Studenten ausgegeben, die gar keine Prüfungen abgelegt haben.

```
<NichtSoGuteStudenten>
  <Student MatrNr="M24002">Xenokrates</Student>
  <Student MatrNr="M26120">Fichte</Student>
  <Student MatrNr="M26830">Aristoxenos</Student>
  <Student MatrNr="M29120">Theophrastos</Student>
  <Student MatrNr="M29555">Feuerbach</Student>
</NichtSoGuteStudenten>
```

Um dies zu verhindern und ausschließlich diejenigen Studenten zu bestimmen, die Prüfungen abgelegt haben, dabei aber nie eine bessere Note als 3.0 hatten, muss nur ein zusätzliches Prädikat in obigen Knotentest aufgenommen werden:

```
<NichtSoGuteStudenten>
{ for $s in doc("Uni.xml")//Student[descendant::Prüfung
    and not(descendant::Prüfung[@Note < 3.0])]
  return
  <Student MatrNr="{ $s/@MatrNr }">
    { $s/Name/text() }
  </Student> }
</NichtSoGuteStudenten>
```

Basierend auf unserer Beispielausprägung ist das Ergebnis der Anfrage leer:

```
<NichtSoGuteStudenten/>
```

- Um den Prüfungsumfang von Studenten zu bestimmen, ermitteln wir für jeden Studenten/jede Studentin die Vorlesungen, über die er oder sie eine Prüfung abgelegt hat. Da wir das Aggregat der Semesterwochenstunden bestimmen müssen, wird die Menge der Vorlesungen \$v im let-Abschnitt der Anfrage gebunden:

```
<Prüfungsumfang>
{ for $s in doc("Uni.xml")/Universität/Studenten/Student
  let $v := doc("Uni.xml")/Universität/Fakultäten/Fakultät/
    ProfessorIn/Vorlesungen/Vorlesung
    [ $s/Prüfungen/Prüfung/@Vorlesung = ./@VorlNr ]
  return
  <StudentMitPrüfungsleistung>
    <Name MatrNr="{ $s/@MatrNr }"> { $s/Name/text() } </Name>
    <PrüfungsSWS> { sum($v/SWS) } </PrüfungsSWS>
  </StudentMitPrüfungsleistung> }
</Prüfungsumfang>
```

Ähnlich zu Teilaufgabe 6 geben wir auch die Studenten aus, die gar keine Prüfungen abgelegt haben:

```

<Prüfungsumfang>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M24002">Xenokrates</Name>
    <PrüfungsSWS>0</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M25403">Jonas</Name>
    <PrüfungsSWS>4</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M26120">Fichte</Name>
    <PrüfungsSWS>0</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M26830">Aristoxenos</Name>
    <PrüfungsSWS>0</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M27550">Schopenhauer</Name>
    <PrüfungsSWS>4</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M28106">Carnap</Name>
    <PrüfungsSWS>4</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M29120">Theophrastos</Name>
    <PrüfungsSWS>0</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
  <StudentMitPrüfungsleistung>
    <Name MatrNr="M29555">Feuerbach</Name>
    <PrüfungsSWS>0</PrüfungsSWS>
  </StudentMitPrüfungsleistung>
</Prüfungsumfang>

```

8. Paare von Studenten und Professoren mit gleichen Namen bestimmen wir, indem wir alle Studenten mit allen Professoren vergleichen (formuliert im for-Abschnitt, der zwei geschachtelten Schleifen entspricht). Die Bedingung der Namensgleichheit überprüfen wir im where-Teil mittels der Funktion **contains(a,b)**, die auf Zeichenketten arbeitet und überprüft, ob **b** ein Substring von **a** ist:

```

<GleicherName>
{ for $s in doc("Uni.xml")/Universität/Studenten/Student,
  $p in doc("Uni.xml")/Universität/Fakultäten/
  Fakultät/ProfessorIn
  where contains(lower-case($s/Name), lower-case($p/Name))
  return
  <Student MatrNr="{ $s/@MatrNr }">

```

```

    {$s/Name}
    <enthaltenerProfName>{$p/Name/text()}</enthaltenerProfName>
  </Student> }
</GleicherName>

```

Führt man die Anfrage auf der Beispielausprägung aus, erhält man ein leeres Element als Ergebnis:

```
<GleicherName/>
```

9. Wir zeigen zuerst eine mögliche Formulierung der Anfrage, die sich aus einer wortgetreuen Umsetzung der Aufgabenstellung ergibt. Das heißt, wir bestimmen zu jedem Professor / jeder Professorin die Studenten, die Vorlesungen bei ihm/ihr gehört haben oder Prüfungen bei ihm/ihr abgelegt haben:

```

<Bekanntheitsgrad>
{ for $p in doc("Uni.xml")//ProfessorIn
  let $bekanntheitsgrad := count(doc("Uni.xml")//Student[
    (some $p in id(../hört/@Vorlesungen)/
      ancestor::ProfessorIn satisfies $p is $p)
    or
    (some $prüfer in id(../@Prüfer) satisfies $prüfer is $p)])
  return
    <ProfessorIn PersNr="{ $p/@PersNr}" Name="{ $p/Name/text()}">
      { $bekanntheitsgrad }
    </ProfessorIn> }
</Bekanntheitsgrad>

```

Damit erhalten wir folgendes Ergebnis:

```

<Bekanntheitsgrad>
  <ProfessorIn PersNr="P2134" Name="Augustinus">2</ProfessorIn>
  <ProfessorIn PersNr="P2136" Name="Curie">0</ProfessorIn>
  <ProfessorIn PersNr="P2127" Name="Kopernikus">0</ProfessorIn>
  <ProfessorIn PersNr="P2125" Name="Sokrates">4</ProfessorIn>
  <ProfessorIn PersNr="P2126" Name="Russel">1</ProfessorIn>
  <ProfessorIn PersNr="P2133" Name="Popper">1</ProfessorIn>
  <ProfessorIn PersNr="P2137" Name="Kant">5</ProfessorIn>
</Bekanntheitsgrad>

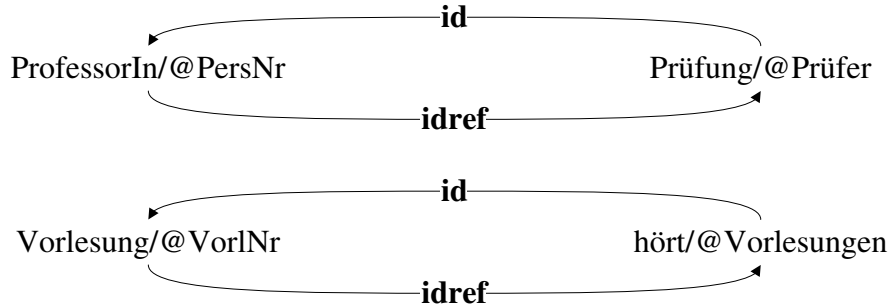
```

Eine existentielle Abhängigkeit wird in XQuery beispielsweise durch **some** ausgedrückt. Informell ist die Bedingung

```
(some $prüfer in id(../@Prüfer) satisfies $prüfer is $p)
```

so zu lesen: "Es existiert ein Knoten \$prüfer auf den ein Prüfer-Attribut (eines Prüfung-Elements) verweist, der identisch mit dem aktuell betrachteten ProfessorIn-Elementknoten \$p ist."

Durch die Funktion **id** werden also die IDREF-Verweise von Prüfer-Attributen auf Professoren aufgelöst. Dasselbe gilt für die Verweise von hört auf Vorlesungen. Folgende Abbildung veranschaulicht den Sachverhalt schematisch:



Alternativ kann eine semantisch äquivalente Anfrage formuliert werden, die diese Beziehungen in der entgegengesetzten Richtung verfolgt, d.h. von ProfessorIn nach Prüfung und von Vorlesung nach hört. Dazu bedienen wir uns der **idref** Funktion. Wir bestimmen so die Studenten, die eine Vorlesung von \$p hören oder von \$p geprüft wurden. Durch **union** fassen wir beide Mengen so zusammen, dass keine Duplikate auftreten:

```
<Bekanntheitsgrad>
{ for $p in doc("Uni.xml")//ProfessorIn
  let $s := (idref($p/Vorlesungen/Vorlesung/@VorlNr)
            [parent::hört]/ancestor::Student union
            idref($p/@PersNr) [parent::Prüfung]/ancestor::Student)
  return
    <ProfessorIn PersNr="{ $p/@PersNr}" Name="{ $p/Name/text()}">
      { count($s) }
    </ProfessorIn> }
</Bekanntheitsgrad>
```

## Aufgabe 2

**Formulierung mittels XPath** Mittels XPath lassen sich beide Anfragen wie folgt formulieren:

```
doc("Uni.xml")//Fakultät[(./Vorlesung)[x]]/FakName
```

```
doc("Uni.xml")//Fakultät[(./Vorlesung)[x]
  and not((./Vorlesung)[x+1])]/FakName
```

**Formulierung mittels XQuery** Dieselben Anfragen können in XQuery elegant über die Aggregatfunktion **count** mit einem = bzw. ≥ Vergleich formuliert werden:

```
<FakultätenMitxVorlesungen>
{ for $f in doc("./Uni.xml")//Fakultät
  let $v := $f//Vorlesung
  where count($v)>=x
  return
    $f/FakName }
</FakultätenMitxVorlesungen>
```

Ruft man die XQuery-Anfrage mit  $x = 5$  auf, so erhält man für unsere Beispielausprägung der Universitätsverwaltung folgendes Ergebnisdokument:

```
<FakultätenMitxVorlesungen>  
  <FakName>Philosophie</FakName>  
</FakultätenMitxVorlesungen>
```