TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe23

Alice Rey, Maximilian Bandle, Michael Jungmair (i3erdb@in.tum.de)
http://db.in.tum.de/teaching/ss23/impldb/

Blatt Nr. 11

Hinweise Die Aufgaben können auf `http://xquery.db.in.tum.de/` getestet werden. Die Daten für das Unischema können mit doc('uni2') geladen werden. Zur Lösung der Aufgaben können Sie die folgenden XQuery-Funktionen verwenden:

max(NUM), count(X), tokenize(STR,SEP), sum(NUM), contains(HAY,NEEDLE)

1. max(NUMBERS) - Returns largest number from list
2. count(LIST) - Return the number of elements in the list
3. tokenize(STR,SEP) - Splits up the string at the seperator
4. sum(NUMBERS) - Returns sum of all numbers in list
5. contains(HAY,NEEDLE) - Checks if the search string (NEEDLE) is contained in the string (HAY)
6. distinct-values(LIST) - Returns the distinct values from the list

Hausaufgabe 1

Lösen Sie in **reinem XPath** folgende Aufgaben und testen Sie diese auf `xquery.db.in.tum.de`.

1. Lassen Sie sich das gesamte Schema anzeigen.

```
doc('uni')
```

2. Finden Sie die Namen aller Fakultäten.

```
doc('uni')//FakName
```

3. Finden Sie die Namen aller Studenten, die Vorlesungen hören.

```
doc('uni')//Student[./hoert]/Name
```

Hausaufgabe 2

Formulieren Sie die zuvor in SQL bearbeiteten Anfragen zur Universitätsdatenbank in XQuery. Erstellen Sie insbesondere XQuery-Anfragen, um folgende Fragestellungen zu beantworten:

a) Suchen Sie die Professoren, die Vorlesungen halten.

```
doc('uni2')//ProfessorIn[.//Vorlesung]/Name
```

b) Finden Sie die Studenten, die alle Vorlesungen gehört haben.

```
doc('uni2')//Student[count(tokenize(hoert/@Vorlesungen," "))=
count(//Vorlesung)]/Name
```

c) Finden Sie die Studenten mit der größten Semesterzahl unter Verwendung von Aggregatfunktionen.

```
let $maxsws:=max(data(doc('uni2')//Student/Semester))
return doc('uni2')//Student[Semester=$maxsws]
(: alternativ als Einzeiler :)
return doc('uni2')//Student[Semester=max(//Student/Semester))]
```

d) Berechnen Sie die Gesamtzahl der Semesterwochenstunden, die die einzelnen Professoren erbringen. Dabei sollen auch die Professoren berücksichtigt werden, die keine Vorlesungen halten.

```
for $p in doc('uni2')//ProfessorIn
return <Prof>{$p/Name}<Summe>{sum(data($p//SWS))}</Summe></Prof>
```

e) Finden Sie die Studenten, die alle vierstündigen Vorlesungen gehört haben.

```
let $fourcount:=count(doc('uni2')//Vorlesung[SWS=4])
for $s in doc('uni2')//Student
where count(
  for $h in tokenize($s/hoert/@Vorlesungen," ")
  where doc('uni2')//Vorlesung[@VorlNr=$h and SWS=4]
  return $h
) = $fourcount
return $s/Name
```

f) Finden Sie die Namen der Studenten, die in keiner Prüfung eine bessere Note als 3.0 hatten.

```
for $s in doc('uni')//Student
where count(
  for $p in $s//Pruefung
  where $p/@Note < 3
  return $p
) = 0
return $s
```

g) Berechnen Sie den Umfang des Prüfungsstoffes jedes Studenten. Es sollen der Name des Studenten und die Summe der Semesterwochenstunden der Prüfungsvorlesungen ausgegeben werden.

```
for $s in doc('uni')//Student
return <Student>{$s/Name}<sum>{
  sum(for $p in $s//Pruefung
      return doc('uni')//Vorlesung[@VorlNr=$p/@Vorlesung]/SWS)}
</sum></Student>
```

h) Finden Sie Studenten, deren Namen den eines Professors enthalten.

```
for $s in doc('uni')//Student
where doc('uni')//ProfessorIn[contains($s/Name,Name)]
return $s/Name
```

i) Ermitteln Sie den Bekanntheitsgrad der Professoren unter den Studenten, wobei wir annehmen, dass Studenten die Professoren nur durch Vorlesungen oder Prüfungen kennen lernen.

```
for $p in doc('uni')//ProfessorIn
return
  <Professor>
    {$p/Name}
    <Bekanntheit>
    {
      count(
        doc('uni')//Student[.//Pruefung[@Pruefer=$p/@PersNr]]/Name
        union
        ( for $v in $p//Vorlesung/@VorlNr
          return doc('uni')//Student[contains(hoert/@Vorlesungen,$v)]/Name)
      )
    }
    </Bekanntheit>
  </Professor>
```

Hausaufgabe 3

Lösen Sie mit XQuery folgende Anfragen und testen Sie diese auf `xquery.db.in.tum.de`.

1. Geben Sie eine nach Rang sortierte Liste der Professoren aus (C4 oben).

```
<Professoren>
{
   for $p in doc('uni')//ProfessorIn
      order by $p/Rang descending
      return $p
}
</Professoren>
```

2. Finden Sie die Namen der Professoren, die die meisten Assistenten haben.

```
<ProfMitAssistenten>
{
    let $maxAssi := max(
      for $p in doc('uni')//ProfessorIn
        return count($p//Assistent)
      )
    return doc('uni')//ProfessorIn[count(.//Assistent)=$maxAssi]/Name
}
</ProfMitAssistenten>
```

3. Finden Sie für jede von einem Studenten gehörte Prüfung den Namen des Prüfers und den Titel der Vorlesung.

```
<Studenten> {
   let $pr := doc('uni')//Assistent union doc('uni')//ProfessorIn
   for $s in doc('uni')//Student
      return <Student>
      {$s/Name}
      <Pruefungen>
      {
      for $p in $s//Pruefung
         let $prName := $pr[./@PersNr=$p/@Pruefer]/Name
         let $vlTitel := doc('uni')
            //Vorlesung[./@VorlNr=$p/@Vorlesung]/Titel
         return <Pruefung Pruefer="{$prName}">{$vlTitel}</Pruefung>
      }
      </Pruefungen>
      </Student>
} </Studenten>
```

Hausaufgabe 4

Geben Sie ein Vorlesungsverzeichnis aus, welches nach dem Umfang der Vorlesungen in SWS gruppiert ist.

Die Ausgabe Ihrer Anfrage soll wie folgt aufgebaut sein:

```
<Vorlesungsverzeichnis>
  <Vorlesungen SWS="2">
    <Vorlesung VorlNr="V5216" Titel="Bioethik"/>
    <Vorlesung VorlNr="V5259" Titel="Der Wiener Kreis"/>
    <Vorlesung VorlNr="V5022" Titel="Glaube und Wissen"/>
    <Vorlesung VorlNr="V5049" Titel="Maeeutik"/>
  </Vorlesungen>
  <Vorlesungen SWS="3">
    <Vorlesung VorlNr="V5043" Titel="Erkenntnistheorie"/>
    <Vorlesung VorlNr="V5052" Titel="Wissenschaftstheorie"/>
  </Vorlesungen>
  <Vorlesungen SWS="4">
    <Vorlesung VorlNr="V4630" Titel="Die 3 Kritiken"/>
    <Vorlesung VorlNr="V5041" Titel="Ethik"/>
    <Vorlesung VorlNr="V5001" Titel="Grundzuege"/>
    <Vorlesung VorlNr="V4052" Titel="Logik"/>
  </Vorlesungen>
</Vorlesungsverzeichnis>
```

```
<Vorlesungsverzeichnis>
{
  for $sws in distinct-values(doc('uni2')//SWS)
  order by $sws
  return
  <Vorlesungen SWS="{$sws}">
  {
    for $vl in doc('uni2')//Vorlesung[SWS=$sws]
    order by $vl/Titel
    return <Vorlesung VorlNr="{$vl/@VorlNr}" Titel="{$vl/Titel}" />
  }
  </Vorlesungen>
}
</Vorlesungsverzeichnis>
```

Hausaufgabe 5

Schreiben Sie eine Anfrage, die folgendes Ergebnis zurückgibt:

```
<Universitaet>
  <Fakultaet Name="Philosophie" AnzahlAssistenten="3">
    <Professor Name="Sokrates" AnzahlAssistenten="2"/>
    <Professor Name="Russel" AnzahlAssistenten="1"/>
  </Fakultaet>
  <Fakultaet Name="Physik" AnzahlAssistenten="2">
    <Professor Name="Kopernikus" AnzahlAssistenten="2"/>
  </Fakultaet>
  <Fakultaet Name="Theologie" AnzahlAssistenten="1">
    <Professor Name="Augustinus" AnzahlAssistenten="1"/>
  </Fakultaet>
</Universitaet>
```

```
<Universitaet>
{for $f in doc('uni')//Fakultaet
    let $fa := count($f//Assistent)
    order by $fa descending
        return <Fakultaet Name="{$f/FakName}" AnzahlAssistenten="{$fa}">{
            for $p in $f//ProfessorIn
            let $pa := count($p//Assistent)
            where $pa > 0
            order by $pa descending
        return <Professor Name="{$p/Name}" AnzahlAssistenten="{$pa}" />
    }</Fakultaet>
}
</Universitaet>
```

Hausaufgabe 6

Datenbanksysteme erlauben JSON-Objekte eingebettet als Attribute in Tabellen. Der zugehörige Syntax ist seit 2017 standardisiert[1] und zum Beispiel in PostgreSQL integriert[2]. Das nachfolgende Statement erstellt eine Hilfstabelle, die einen Ausschnitt des Uni-Schemas als JSON-Objekt enthält (und lässt sich in `hyper-db.de` eingeben).

[1] `https://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip`
[2] `https://www.postgresql.org/docs/current/functions-json.html`

```
with uni_json (name, doc) as (values ('VirtU', '{
  "Name": "Virtuelle Universitaet der Grossen Denker",
  "UniLeitung": {"Rektor": "Sokrates", "Kanzler": "Erhard"},
  "Fakultaeten":  [
    { "Name": "Philosophie", "Professoren": [
        { "PersNr": 2125, "Name": "Sokrates", "Rang": "C4",
          "Vorlesungen": [ {"VorlNr": 5041, "Titel": "Ethik", "SWS": 4},
            {"VorlNr": 5049, "Titel": "Maeeutik", "SWS": 2},
            {"VorlNr": 4052, "Titel": "Logik", "SWS": 4}]
        }]}]}'::json))
```

1. Geben Sie in SQL den Namen der jeweils ersten Fakultät in `uni_json` aus.

   ```
   select doc->'Fakultaeten'->0 from uni_json
   ```

2. Geben Sie in SQL die Personalnummer (`PersNr`) des ersten Professors der jeweils ersten Fakultät aus.

   ```
   select doc->'Fakultaeten'->0->'Professoren'->0->'PersNr' from uni_json
   ```

3. Joinen Sie diese mit der SQL-Relation `pruefen` und `Studenten`, um die Namen aller von ihm geprüften Studenten auszugeben.

   ```
   select s.Name from uni_json, pruefen p, Studenten s
   where cast(doc->'Fakultaeten'->0->'Professoren'->0->>'PersNr' as int) =
   p.PersNr and p.MatrNr=s.MatrNr;
   ```