



Übung zur Vorlesung *Einsatz und Realisierung von Datenbanken* im SoSe20

Maximilian {Bandle, Schüle}, Josef Schmeißer (i3erdb@in.tum.de)

<http://db.in.tum.de/teaching/ss20/impldb/>

Blatt Nr. 11

Hausaufgabe 1

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 Student gehörte Prüfung den Namen des Prüfers und 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 2

Geben Sie ein Vorlesungsverzeichnis aus, welches nach dem Umfang der Vorlesungen in SWS gruppiert ist ¹.

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 3

Schreiben Sie eine Anfrage, die folgendes zurück gibt:

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

¹Sie können die Aufgabe unter <http://xquery.db.in.tum.de> mit dem doc('uni2') Datensatz testen.

```

    <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 4

Datenbanksysteme erlauben JSON-Objekte eingebettet als Attribute in Tabellen. Der zugehörige Syntax ist seit 2017 standardisiert² und zum Beispiel in PostgreSQL integriert³. Das nachfolgende Statement erstellt eine Hilfstabelle, die einen Ausschnitt des Uni-Schemas als JSON-Objekt enthält (und lässt sich in `hyper-db.de` eingeben).

```

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;
```

²https://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip

³<https://www.postgresql.org/docs/current/functions-json.html>

Hausaufgabe 5

Vervollständigen Sie die untere Anfrage um die Namen der Freunde von Personen mit dem Vornamen *Sokrates* zu finden, die älter als 30 Jahre sind. Die *foaf* Ontology is unter <http://xmlns.com/foaf/spec/> beschrieben. Nutzen Sie <https://rdf.db.in.tum.de/> für Ihre Abfrage.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name2
WHERE {
    . . . . .
}
```

Lösung:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name2
WHERE {
    ?person1 foaf:knows ?person2 .
    ?person1 foaf:firstName "Sokrates" .
    ?person2 foaf:name ?name2 .
    ?person22 foaf:name ?name2 .
    ?person22 foaf:age ?age .
    FILTER ( ?age > 30 )
}
```

Hausaufgabe 6

```
@prefix ex: <http://example.org>.
ex:Rapunzel ex:hatAutor ex:Sokrates.
ex:Rapunzel ex:erschienen 2006.
ex:Aschenputtel ex:hatAutor ex:Archimedes.
ex:Aschenputtel ex:hatAutor ex:Platon.
ex:Schneewittchen ex:hatAutor ex:Platon.
ex:Schneewittchen ex:erschienen 2004.
```

Drücken Sie die folgenden Anfragen in SPARQL aus:

1. Geben Sie alle Bücher aus, für die sowohl der Autor als auch das Erscheinungsjahr in der Datenbank enthalten sind.

```
PREFIX ex: <http://example.org>
SELECT ?book
WHERE {
    ?book ex:erschienen ?jahr .
    ?book ex:hatAutor ?autor .
}
```

2. Geben Sie die gemeinsamen Autoren der beiden Bücher Aschenputtel und Schneewittchen aus.

```
PREFIX ex: <http://example.org>
SELECT ?autor
WHERE {
    ex:Aschenputtel ex:hatAutor ?autor .
    ex:Schneewittchen ex:hatAutor ?autor .
}
```

3. Geben Sie die Namen aller Autoren (ohne Duplikate) von Büchern mit einem Erscheinungsjahr nach 2004 aus.

```
PREFIX ex: <http://example.org>
SELECT DISTINCT ?autor
WHERE {
    ?book ex:hatAutor ?autor .
    ?book ex:erschiene ?jahr.
    FILTER ( ?jahr > 2004 )
}
```