TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Exercises for *Foundations in Data Engineering*, WiSe 22/23

Alexander Beischl, Maximilian Reif (i3fde@in.tum.de)
http://db.in.tum.de/teaching/ws2223/foundationsde

Sheet Nr. 13

Exercise 1 RDF Query Optimization

a) Given the following RDF triples, determine the set $\mathcal{S}_C$ of characteristic sets $S_C(s)$ for subjects s.

S	P	O
Catch Me If You Can	publication date	2002
Catch Me If You Can	IMDb ID	tt0264464
Leonardo DiCaprio	starred in	Catch Me If You Can
Leonardo DiCaprio	award received	Golden Globe Award for Best Actor
Grace Hopper	award received	Presidential Medal of Freedom
Tom Hanks	starred in	Catch Me If You Can
Tom Hanks	award received	Presidential Medal of Freedom
Tom Hanks	member of political party	Democratic Party
Nathalie Baye	starred in	Catch Me If You Can
Nathalie Baye	award received	César Award for Best Actress

Solution:
Characteristic set of one subject s:
$S_C(s) := \{p | \exists o : (s, p, o) \in R\}$
Set of characteristic sets:
$\mathcal{S}_C(R) := \{S_C(s) | \exists p, o : (s, p, o) \in R\}$

Therefore the $S_C(s)$ are:

$$\begin{aligned} S_C(\text{Catch Me If You Can}) &= \{\text{pub. date, IMDb ID}\} \\ S_C(\text{Leonardo DiCaprio}) &= \{\text{starred in, award received}\} \\ S_C(\text{Grace Hopper}) &= \{\text{award received}\} \\ S_C(\text{Tom Hanks}) &= \{\text{starred in, award received, member...}\} \\ S_C(\text{Nathalie Baye}) &= \{\text{starred in, award received}\} \end{aligned}$$

Thus $\mathcal{S}_C$:

$$\begin{aligned} &\{\{\text{pub. date, IMDb ID}\}^1 \\ &\{\text{starred in, award received}\}^2 \\ &\{\text{award received}\}^1 \\ &\{\text{starred in, award received, member...}\}^1\} \end{aligned}$$

b) Use the characteristic set to estimate the result cardinality of this query:

```
select distinct ?a
where {
  ?a <starred in> ?m.
  ?a <award received> ?n.
}
```

Solution:
Estimate cardinality with:
$\Sigma_{S \in \{S | S \in \mathcal{S}_C(R) \wedge \{starred\ in, award\ received\} \subseteq S} count(S)$
supersets of $\{starredin, awardreceived\}$:

$$\begin{aligned} S_C(\text{Leonardo DiCaprio}) &= \{\text{starred in, award received}\} \\ S_C(\text{Tom Hanks}) &= \{\text{starred in, award received, member...}\} \\ S_C(\text{Nathalie Baye}) &= \{\text{starred in, award received}\} \end{aligned}$$

Therefore the estimation is 3.

Exercise 2 Solve the following tasks by using XQuery. You can test your queries at: `http://xquery.db.in.tum.de` by using the doc('uni3') dataset.

a) Return all professors who give one or more lectures.
Solution:
```
doc('uni3')//professor[.//lecture]/name
```

b) Find all students who attend all lectures.
Solution:
```
doc('uni3')//student[count(tokenize(attends/@lectures," "))=
count(//lecture)]/name
```

c) Return students with the highest number of semesters, use aggregate functions.
Solution:
```
let $maxsem:=max(data(doc('uni3')//student/semester))
return doc('uni3')//student[semester=$maxsem]
```

d) Calculate the sum of semester periods per week (sws) for each professor. Include professors without any lecture in your results.
Solution:
```
for $p in doc('uni3')//professor
return <prof>{$p/name}<sum>{sum(data($p//sws))}</sum></prof>
```

e) Find all students attending all four-hour lectures.

Solution:

```
let $fourcount:=count(doc('uni3')//lecture[sws=4])
for $s in doc('uni3')//student
where count(
  for $l in tokenize($s/attends/@lectures," ")
  where doc('uni3')//lecture[@lectureNr=$l and sws=4]
  return $l
) = $fourcount
return $s/name
```

f) Return the name of all students who never had a grade better than 3.0 in any exam.

Solution:

```
for $s in doc('uni3')//student
where count(
  for $e in $s//exam
  where $e/@grade < 3
  return $e
) = 0
return $s
```

g) Calculate the examination material's extent for each student. Therefore, return the name of the student and the sum of semester periods per week of all lectures in which the student wrote an exam.

Solution:

```
for $s in doc('uni3')//student
return <student>{$s/name}<sum>{
  sum(for $e in $s//exam
      return doc('uni3')//lecture[@lectureNr=$e/@lecture]/sws)}
</sum></student>
```

h) Some of the students' names contain the name of a professor. Return all such students.

Solution:

```
for $s in doc('uni3')//student
  where doc('uni3')//professor[contains($s/name,name)]
  return $s/name
```

i) Determine the level of awareness of each professor among students. We assume students only get to know professors by attending their lectures or writing an exam.

Solution:

```
for $p in doc('uni3')//professor
return
  <professor>
    {$p/name}
    <level_of_awareness>
    {
      count(
        doc('uni3')//student[.//exam[@examiner=$p/@persNr]]/@matrNr
        union
        ( for $lnr in $p//lecture/@lectureNr
          return doc('uni3')//student[contains(attends/@lectures,$lnr)]/@matrNr)
      )
    }
    </level_of_awareness>
  </professor>
```

Exercise 3 The queries in this exercise work on the mondial dataset

```
http://dbis.informatik.uni-goettingen.de/Mondial/mondial.dtd
http://dbis.informatik.uni-goettingen.de/Mondial/mondial.xml
```

For the execution of XQuery requests, you may use xqilla for example:

```
apt-get install xqilla
```

The example in exercise 1 assumes, that the `mondial.*` files are in the current working directory.

1. Set up an XQuery engine and output the whole mondial document.

   Example:

   ```
   let $mondial := doc("mondial.xml")/mondial
   return $mondial
   ```

2. Find the names of all continents.
   Solution:

   ```
   let $mondial := doc("mondial.xml")/mondial
   return $mondial/continent/name/text()
   ```

3. For every country, compute the average population of its cities.

   ```
   <country name="Albania">79166.66666666667</country>
   ...
   <country name="Finland">100842.11764705883</country>
   ...
   ```

   Solution:

   ```
   let $mondial := doc("mondial.xml")/mondial
      for $count in $mondial/country
         return <country name="{$count/name}">
           {avg($count//city/population[last()])}
           </country>
   ```

4. For every country, compute the percentage of population living in cities to the whole population.

```
<country name="Albania">14.619271092376557</country>
<country name="Greece">18.632551932449434</country>
...
```

Solution:

```
let $mondial := doc("mondial.xml")/mondial
for $count in $mondial/country
   where $count/population
   return <country name="{$count/name}">
     {sum($count//city/population[last()]) * 100 div $count/population[last()]}
   </country>
```

5. Find all continents with their names and countries in the continents.

```
<continent name="Europe">
    <country name="Albania"/>
    <country name="Greece"/>...
    ...
</continent>
...
```

Solution:

```
let $mondial := doc("mondial.xml")/mondial
for $cont in $mondial/continent
   return <continent name="{$cont/name}"> {
      for $count in $mondial/country[encompassed/@continent=$cont/@id]
         return <country name="{$count/name}"/>
   } </continent>
```