1. Unit: SPARQL

Exercise 1.1 (SPARQL-Queries) Give SPARQL queries against `mondial.n3` that yield answers to the following questions:

- Names and populations (ordered) of all countries that have more than 10.000.000 inhabitants.
- Names of all countries that have at least one city with more than 1.000.000 inhabitants.
- Names of all countries that have no city with more than 1.000.000 inhabitants.
- Names of all european countries that have no membership in the European Union.
- Abbreviations of all organizations whose headquarter is located in the capital of a member country (together with the names of the country and the city).

```
# bigcountries.sparql
prefix mon: <http://www.semwebtech.org/mondial/10/meta#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?N ?P
FROM <file:mondial.n3>
WHERE {?X rdf:type mon:Country . ?X mon:name ?N . ?X mon:population ?P .
       FILTER (?P > 10000000) }
ORDER BY DESC(?P)
```

```
# bigcities.sparql
prefix mon: <http://www.semwebtech.org/mondial/10/meta#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?N
FROM <file:mondial.n3>
WHERE {?X rdf:type mon:Country . ?X mon:name ?N .
       ?X mon:hasCity ?C . ?C mon:population ?P .
       FILTER (?P > 1000000) }
```

```
# nobigcities.sparql
prefix mon: <http://www.semwebtech.org/mondial/10/meta#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT distinct ?N
FROM <file:mondial.n3>
WHERE {?X rdf:type mon:Country . ?X mon:name ?N .
       OPTIONAL { ?X mon:hasCity ?C . ?C mon:population ?P . FILTER (?P > 1000000) } .
       FILTER (!BOUND(?P)) }
```

```
# no-eu.sparql
prefix mon: <http://www.semwebtech.org/mondial/10/meta#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT distinct ?N
FROM <file:mondial.n3>
WHERE {?X rdf:type mon:Country . ?X mon:name ?N .
       ?X mon:encompassed [ mon:name 'Europe' ] .
```

```
       OPTIONAL { ?X mon:isMember [ mon:name 'European Union'] .
                  ?X mon:name ?XX }
       FILTER (!BOUND(?XX)) }
# bind XX (to anything ...) in case that EU membership is satisfied
```

```
# cap-hq.sparql
prefix mon: <http://www.semwebtech.org/mondial/10/meta#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?XN ?OA ?CN
FROM <file:mondial.n3>
WHERE {?O rdf:type mon:Organization . ?O mon:abbrev ?OA .
       ?O mon:hasHeadq ?C .
       ?X rdf:type mon:Country . ?X mon:carCode ?XN .
       ?X mon:capital ?C . ?C mon:name ?CN .
       ?X mon:isMember ?O }
ORDER BY ?CN
```

This is an example for a *cyclic join*: Organization - hasHeadq - City - isCapital - Country - isMember - Organization. Note the occurrence of the join variable O that closes this circle. When evaluated in the same order as stated in the query, the last triple pattern acts as a selection (the actual evaluation order is defined by the optimizer).

Exercise 1.2 (SPARQL Optional) Give a SPARQL query against `mondial.n3` that yield answers to the following question:

- For each country, give the name, and the population.
  If more than 1/4 of the population are living in its capital, give also the name and the population of the capital.
- Give the same query in SQL (against relational Mondial) and in XML/XQuery (against mondial.xml).

```
# country-cap-quarterpop.sparql
prefix mon: <http://www.semwebtech.org/mondial/10/meta#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select ?N1 ?P1 ?N2 ?P2
from <file:mondial.n3>
where { ?X a mon:Country; mon:name ?N1; mon:population ?P1 .
        OPTIONAL {?X mon:capital [ mon:name ?N2; mon:population ?P2 ]
                  FILTER (?P2 > 0.25 * ?P1) }}
```

The central issue of this exercise is the "if-"functionality of the OPTIONAL with a filter in it.

The same in XQuery can explicitly use XQuery's *functional-style* if-construct:

```
for $c in //country
let $pop := $c/population[position()=last()],
    $cap := id($c/@capital),
    $cappop := $cap/population[position()=last()]
return
```

```
<result code="{$c/@car_code}" pop="{$c/population[position()=last()]}">
 { if ($cappop > 0.25 * $pop)
   then ( attribute{"cap"}{$cap/name}, attribute{"cappop"}{$cappop} )
   else ()
 }
</result>
```

The algebraically closest SQL relative to SPARQL's OPTIONAL is the outer join (note that the outer SELECT-FROM is semantically redundant, but syntactically required; cf. that for the UNION operator it would not be necessary)

```
SELECT x.code, x.population, y.name, y.population
FROM
 (SELECT code, population
  FROM country) x
 LEFT OUTER JOIN
 (SELECT country.code, city.name, city.population
  FROM city, country
  WHERE city.name=country.capital AND city.country=country.code
    AND city.province= country.province
    AND city.population > 0.25 * country.population) y
 ON x.code = y.code
```

Another way with classical SQL is a UNION (note the OR-NULL check in the second subquery):

```
(SELECT x.code, x.population, y.name, y.population
 FROM Country x, City y
 WHERE x.capital = y.name AND x.code=y.country AND x.province=y.province
   AND y.population > 0.25* x.population)
UNION
(SELECT x.code, x.population, NULL, NULL
 FROM Country x, City y
 WHERE x.capital = y.name AND x.code=y.country AND x.province=y.province
   AND (y.population IS NULL
        OR NOT(y.population > 0.25* x.population)))
```

SQL today also supports the functional CASE-WHEN-THEN-ELSE construct:

```
SELECT x.code, x.population,
  CASE WHEN y.population > 0.25* x.population THEN y.name ELSE NULL END AS cap,
  CASE WHEN y.population > 0.25* x.population THEN y.population ELSE NULL END AS cappop
FROM Country x, City y
WHERE x.capital = y.name AND x.code=y.country AND x.province=y.province
```

It is important to know which constructs a language supports and when and how to use them.