Practical on SPARQL

Semantic Web module

RDF Dataset - Library

- Consider the following RDF dataset

```
@prefix library:      <http://example.org/library#>.
@prefix book:         <http://example.org/book#>.

library:A_Semantic_Web_Primer_1 a library:Book;
                      book:isbn "978-0262012102";
                      book:publicationYear 2004;
                      book:pages 272;
                      book:writtenBy library:G_Antoniou,
                                     library:F_van_Harmelen.
library:A_Semantic_Web_Primer_2 a library:Book;
                      book:isbn "978-0262012423";
                      book:publicationYear 2008;
                      book:writtenBy library:G_Antoniou,
                                     library:F_van_Harmelen.
```

RDF Dataset - Library (cont'd)

```
library:A_Semantic_Web_Primer_3 a library:Book;
                    book:isbn "978-0262018289";
                    book:publicationYear 2012;
                    book:pages 288;
                    book:writtenBy library:G_Antoniou,
                                   library:F_van_Harmelen,
                                   library:P_Groth,
                                   library:R_Hoekstra.
library:G_Antoniou a library:Author;
             library:name "Grigoris Antoniou".
library:F_van_Harmelen a library:Author;
             library:name "Frank van Harmelen".
library:P_Groth a library:Author;
             library:name "Paul Groth".
library:R_Hoekstra a library:Author;
             library:name "Rinke Hoekstra".
```

SPARQL Queries – Library (1/16)

- All SPARQL queries for the library domain start with:

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>
```

SPARQL Queries – Library (2/16)

- The books that are represented in the dataset.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book
WHERE {
      ?book a library:Book.
}
```

?book
library:A_Semantic_Web_Primer_1
library:A_Semantic_Web_Primer_2
library:A_Semantic_Web_Primer_3

SPARQL Queries – Library (3/16)

- The books and their authors.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book ?author
WHERE {
       ?book a library:Book.
       ?book book:writtenBy ?author.
}
```

SPARQL Queries – Library (4/16)

- The books and their authors (results).

?book	?author
library:A_Semantic_Web_Primer_1	library:G_Antoniou
library:A_Semantic_Web_Primer_1	library:F_van_Harmelen
library:A_Semantic_Web_Primer_2	library:G_Antoniou
library:A_Semantic_Web_Primer_2	library:F_van_Harmelen
library:A_Semantic_Web_Primer_3	library:G_Antoniou
library:A_Semantic_Web_Primer_3	library:F_van_Harmelen
library:A_Semantic_Web_Primer_3	library:P_Groth
library:A_Semantic_Web_Primer_3	library:R_Hoekstra

SPARQL Queries – Library (5/16)

- The books that the name of their authors is "Rinke Hoekstra".

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book
WHERE {
      ?book a library:Book.
      ?book book:writtenBy ?author.
      ?author library:name "Rinke Hoekstra"^^xsd:string.
}
```

?book
library:A_Semantic_Web_Primer_3

SPARQL Queries – Library (6/16)

- The books that are written by Grigoris Antoniou or Frank van Harmelen.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book
WHERE {
      {?book book:writtenBy library:G_Antoniou.}
      UNION
      {?book book:writtenBy library:F_van_Harmelen.}
}
```

SPARQL Queries – Library (7/16)

- The books that are written by Grigoris Antoniou or Frank van Harmelen (results).

?book
library:A_Semantic_Web_Primer_2
library:A_Semantic_Web_Primer_1
library:A_Semantic_Web_Primer_3
library:A_Semantic_Web_Primer_2
library:A_Semantic_Web_Primer_3
library:A_Semantic_Web_Primer_1

- Some books are retuned twice because they were written by both Grigoris Antoniou and Frank van Harmelen.
- If result list is not explicitly ordered, books are returned in any (generally unpredictable) order.

SPARQL Queries – Library (8/16)

- The books that are written by Grigoris Antoniou or Frank van Harmelen (return each book at most once).

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT DISTINCT ?book
WHERE {
      {?book book:writtenBy library:G_Antoniou.}
      UNION
      {?book book:writtenBy library:F_van_Harmelen.}
}
```

?book
library:A_Semantic_Web_Primer_2
library:A_Semantic_Web_Primer_1
library:A_Semantic_Web_Primer_3

SPARQL Queries – Library (9/16)

- At most two books that were written by Grigoris Antoniou or Frank van Harmelen.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT DISTINCT ?book
WHERE {
      {?book book:writtenBy library:G_Antoniou.}
      UNION
      {?book book:writtenBy library:F_van_Harmelen.}
}
LIMIT 2
```

?book
library:A_Semantic_Web_Primer_2
library:A_Semantic_Web_Primer_1

SPARQL Queries – Library (10/16)

- The books and if known, their number of pages.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book ?pages
WHERE {
      ?book a library:Book.
      OPTIONAL
      {?book book:pages ?pages.}
}
```

?book	?pages
library:A_Semantic_Web_Primer_1	272
library:A_Semantic_Web_Primer_2	
library:A_Semantic_Web_Primer_3	288

SPARQL Queries – Library (11/16)

- The books that were published after 2010.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book
WHERE {
      ?book a library:Book.
      ?book book:publicationYear ?year.
      FILTER (?year>2010).
}
```

?book
library:A_Semantic_Web_Primer_3

SPARQL Queries – Library (12/16)

- The books that were published between 2002 and 2009.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book
WHERE {
      ?book a library:Book.
      ?book book:publicationYear ?year.
      FILTER (?year>=2002 && ?year<=2009).
}
```

?book
library:A_Semantic_Web_Primer_1
library:A_Semantic_Web_Primer_2

SPARQL Queries – Library (13/16)

- The authors with name that starts with "Grigoris".

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?author
WHERE {
      ?author a library:Author.
      ?author library:name ?name.
      FILTER regex(?name,"^Grigoris").
}
```

?author
library:G_Antoniou

SPARQL Queries – Library (14/16)

- The authors with name that contains "o".

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?author
WHERE {
      ?author a library:Author.
      ?author library:name ?name.
      FILTER regex(?name,"o").
}
```

?author
library:R_Hoekstra
library:P_Groth
library:G_Antoniou

SPARQL Queries – Library (15/16)

- The books, their isbn and their publication year ordered (in descending order) by their publication year.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT ?book ?isbn ?year
WHERE {
       ?book a library:Book.
       ?book book:isbn ?isbn.
       ?book book:publicationYear ?year.
}
ORDER BY DESC(?year)
```

?book	?isbn	?year
library:A_Semantic_Web_Primer_3	978-0262018289	2012
library:A_Semantic_Web_Primer_2	978-0262012423	2008
library:A_Semantic_Web_Primer_1	978-0262012102	2004

SPARQL Queries – Library (16/16)

- The number of books that are written by Grigoris Antoniou.

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

SELECT (COUNT(?book) AS ?numberOfBooks)
WHERE {
      ?book a library:Book.
      ?book book:writtenBy library:G_Antoniou.
}
```

?numberOfBooks
3

RDF Dataset - Geography

- Consider the following RDF dataset

```
@prefix geo:          <http://www.geography.org/schema.rdf#>.
@prefix countries:    <http://example.org/countries#>.
@prefix cities:       <http://example.org/cities#>.

countries:Belgium     a geo:Country;
                      geo:capital cities:Brussels;
                      geo:borders countries:France,
                                  countries:Netherlands,
                                  countries:Germany,
                                  countries:Luxembourg.
```

RDF Dataset - Geography (cont'd)

```
countries:Netherlands a geo:Country;
                      geo:capital cities:Amsterdam;
                      geo:population 16645313;
                      geo:borders countries:Belgium,
                           countries:Germany.
countries:Luxembourg  a geo:Country;
                      geo:capital cities:Luxembourg;
                      geo:population 517000;
                      geo:borders countries:France,
                           countries:Belgium, countries:Germany.
cities:Amsterdam      a geo:City;
                      geo:areaCode "020".
cities:Brussels       a geo:City;
                      geo:areaCode "02".
cities:Luxembourg     a geo:City;
                      geo:areaCode "00352".
```

Exercise 1

- Write appropriate SPARQL queries (and their results) that return
  1. The countries that are represented in the dataset.
  2. The countries and their capitals.
  3. The countries that the area code of their capitals is "020".
  4. The countries that border Germany or France.
  5. At most two countries that border Germany or France.
  6. The countries and if known, their populations.
  7. The capitals of the countries that border France.
  8. The capitals of the countries that border a country that borders France.
  9. The countries with population over 1000000.
  10. The countries with population between 500000 and 900000.
  11. The cities with area code that starts with "02".
  12. The cities with area code that contains "2".

Exercise 1 (cont'd)

13. The countries, their capitals and their populations ordered (in descending order) by their population.
14. The number of countries that border Germany.

Exercise 1 - Solution

- All SPARQL queries for this exercise start with:

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>
```

Exercise 1 - Solution

1. The countries that are represented in the dataset.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country
WHERE {
      ?country a geo:Country.
}
```

? country
countries:Belgium
countries:Netherlands
countries:Luxembourg

Exercise 1 - Solution

2. The countries and their capitals.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country ?capital
WHERE {
      ?country a geo:Country.
      ?country geo:capital ?capital.
}
```

? country	?capital
countries:Belgium	cities:Brussels
countries:Netherlands	cities:Amsterdam
countries:Luxembourg	cities:Luxembourg

Exercise 1 - Solution

3. The countries that the area code of their capitals is "020".

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country
WHERE {
      ?country a geo:Country.
      ?country geo:capital ?capital.
      ?capital geo:areaCode "020"^^xsd:string.
}
```

? country
countries:Netherlands

Exercise 1 - Solution

4. The countries that border Germany or France.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country
WHERE {
      {?country geo:borders countries:Germany.}
      UNION
      {?country geo:borders countries:France.}
}
```

Exercise 1 - Solution

4. The countries that border Germany or France (results)

? country
countries:Belgium
countries:Netherlands
countries:Luxembourg
countries:Belgium
countries:Luxembourg

- Some countries are retuned twice because they border both Germany and France
- If result list is not explicitly ordered, countries are returned in any (generally unpredictable) order

Exercise 1 - Solution

4. The countries that border Germany or France (return each country at most once).

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT DISTINCT ?country
WHERE {
      {?country geo:borders countries:Germany.}
      UNION
      {?country geo:borders countries:France.}
}
```

? country
countries:Belgium
countries:Netherlands
countries:Luxembourg

Exercise 1 - Solution

5. At most two countries that border Germany or France.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT DISTINCT ?country
WHERE {
      {?country geo:borders countries:Germany.}
      UNION
      {?country geo:borders countries:France.}
}
LIMIT 2
```

? country
countries:Belgium
countries:Netherlands

Exercise 1 - Solution

6. The countries and if known, their populations.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country ?population
WHERE {
      ?country a geo:Country.
      OPTIONAL
      {?country geo:population ?population.}
}
```

? country	?population
countries:Belgium	
countries:Netherlands	16645313
countries:Luxembourg	517000

Exercise 1 - Solution

7. The capitals of the countries that border France

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?capital
WHERE {
      ?country a geo:Country.
      ?country geo:capital ?capital.
      ?country geo:borders countries:France.
}
```

? capital
cities:Brussels
cities:Luxembourg

Exercise 1 - Solution

8. The capitals of the countries that border a country that borders France.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?capital
WHERE {
      ?country a geo:Country.
      ?country geo:capital ?capital.
      ?country geo:borders ?anotherCountry.
      ?anotherCountry geo:borders countries:France.
}
```

? capital
cities:Brussels
cities:Amsterdam
cities:Luxembourg

Exercise 1 - Solution

9. The countries with population over 1000000

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country
WHERE {
      ?country a geo:Country.
      ?country geo:population ?population.
      FILTER (?population>1000000).
}
```

? country
countries:Netherlands

Exercise 1 - Solution

10. The countries with population between 500000 and 900000

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country
WHERE {
      ?country a geo:Country.
      ?country geo:population ?population.
      FILTER (?population>500000 && ?population<900000).
}
```

? country
countries:Luxembourg

Exercise 1 - Solution

11. The cities with area code that starts with "02".

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?city
WHERE {
      ?city a geo:City.
      ?city geo:areaCode ?code.
      FILTER regex(?code,"^02").
}
```

? city
cities:Amsterdam
cities:Brussels

Exercise 1 - Solution

12. The cities with area code that contains "2".

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?city
WHERE {
      ?city a geo:City.
      ?city geo:areaCode ?code.
      FILTER regex(?code,"2").
}
```

? city
cities:Amsterdam
cities:Brussels
cities:Luxembourg

Exercise 1 - Solution

13. The countries, their capitals and their populations ordered (in descending order) by their population.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT ?country ?capital ?population
WHERE {
      ?country a geo:Country.
      ?country geo:capital ?capital.
      ?country geo:population ?population.
}
ORDER BY DESC(?population)
```

? country	?capital	?population
countries:Netherlands	cities:Amsterdam	16645313
countries:Luxembourg	cities:Luxembourg	517000

Exercise 1 - Solution

14. The number of countries that border Germany.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

SELECT (COUNT(?country) AS ?numberOfCountries)
WHERE {
      ?country a geo:Country.
      ?country geo:borders countries:Germany.
}
```

? numberOfCountries
3

SPARQL Queries – Library (1/2)

- Write a SPARQL query that checks whether there are authors with name that starts with "Grigoris".

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

ASK
WHERE {
      ?author a library:Author.
      ?author library:name ?name.
      FILTER regex(?name,"^Grigoris").
}
```

SPARQL Queries – Library (2/2)

- Write a SPARQL query that for each book that is written by its author, it creates a statement of the form:

```
?author book:authorOf ?book
```

```
PREFIX library:      <http://example.org/library#>
PREFIX book:         <http://example.org/book#>

CONSTRUCT {?author book:authorOf ?book.}
WHERE {
      ?book a library:Book.
      ?book book:writtenBy ?author.
}
```

Exercise 2

- Write a SPARQL query that checks whether there is any country that borders France.
- Write a SPARQL query that checks whether there is any country with population that is over 20000000.
- Write a SPARQL query that for each country that borders with France, it creates a statement of the form:

```
countries:France geo:borders ?country
```

- Write a SPARQL query that for each country that its population is less than 1000000, it creates a statement of the form:

```
?country geo:size geo:small.
```

Exercise 2 - Solution

- Write a SPARQL query that checks whether there is any country that borders France.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

ASK
WHERE {
      ?country a geo:Country.
      ?country geo:borders countries:France.
}
```

Exercise 2 - Solution

- Write a SPARQL query that checks whether there is any country with population that is over 20000000.

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

ASK
WHERE {
      ?country a geo:Country.
      ?country geo:population ?population.
      FILTER (?population>20000000).
}
```

Exercise 2 - Solution

- Write a SPARQL query that for each country that borders France, it creates a statement of the form:

```
countries:France geo:borders ?country
```

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

CONSTRUCT {countries:France geo:borders ?country.}
WHERE {
      ?country a geo:Country.
      ?country geo:borders countries:France.
}
```

Exercise 2 - Solution

- Write a SPARQL query that for each country that its population is less than 1000000, it creates a statement of the form:

```
?country geo:size "small".
```

```
PREFIX geo:          <http://www.geography.org/schema.rdf#>
PREFIX countries:    <http://example.org/countries#>
PREFIX cities:       <http://example.org/cities#>

CONSTRUCT {?country geo:size geo:small.}
WHERE {
      ?country a geo:Country.
      ?country geo:population ?population.
      FILTER (?population<1000000).
}
```