Semantic Web handout including: lecture questions and practical sessions

In this document, you must provide your answers to the questions asked during the course **and** to the questions of the practical sessions; everything in one document.
The questions of the course have been repeated here; **do not delete the questions** but provide your answer to each question just below the question. You can use screenshots when appropriate as an answer to a question but keep your answers and this file as small and concise as possible.
At the end, you must generate and submit only one final PDF file based on this template.
In questions where you are asked to create, invent or use your own data, make sure they are different from other student's.

First name: Boa Jean-Luc
Family name: BOA THIEMELE
Email: jeanlucthiemele@gmail.com

TABLE OF CONTENTS:

QUESTIONS FROM THE COURSES

Questions from the course on Linked Data.

Q1.1 Practice XML replace missing parts.

```
<archi_book>
 <short_title>Architecture Now</short_title>
 <main_author>Jodidio, Philip</main_author>
 <ID isbn10="3822840912"/>
</archi_book>
```

Q1.2 Provide 10 first lines
Get 10 first lines of the five results for:
http://www.wikidata.org/entity/Q23014205
http://www.wikidata.org/entity/Q23014205.json
http://www.wikidata.org/entity/Q23014205.rdf
http://www.wikidata.org/entity/Q23014205.ttl
http://www.wikidata.org/entity/Q23014205.nt

- http://www.wikidata.org/entity/Q23014205

Fabien Gandon (Q23014205)
From Wikidata
Jump to navigation Jump to search
computer science researcher
edit
Language
Label
Description
Also known as
English
Fabien Gandon
computer science researcher
Statements
instance of
human

- http://www.wikidata.org/entity/Q23014205.json :

{"entities":{"Q23014205":{"pageid":25028548,"ns":0,"title":"Q23014205","lastrevid":2056781113,"modified":"2024-01-18T13:58:43Z","type":"item","id":"Q23014205","labels":{"fr":{"language":"fr","value":"Fabien

- http://www.wikidata.org/entity/Q23014205.rdf :

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:ontolex="http://www.w3.org/ns/lemon/ontolex#" /"
xmlns:ref="http://www.wikidata.org/reference/"

- http://www.wikidata.org/entity/Q23014205.ttl :

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> .

- http://www.wikidata.org/entity/Q23014205.nt :

<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Dataset> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://schema.org/about> <http://www.wikidata.org/entity/Q23014205> Q1.3 DBpedia

1. Find “London” on DBpedia.org; e.g. Google: "london site:dbpedia.org“
   make sure you are on the English chapter (dbpedia.org) as there are many others (fr.dbpedia.org, de.dbpedia.org)
2. Find dbp:populationDemonym and give its value
3. Find rdf:type
4. Click on value yago:WikicatCapitalsInEurope
5. Find “Vienna” and get its URI
   (careful: with content negotiation and redirection, the URL of the page you are currently viewing may be different from the URI of the resource it describes)
6. ISO code of the Vienna region?

dbp:populationDemonyms : Londoner (en)
dbr:Vienna : http://dbpedia.org/resource/Vienna
dbo:isoCodeRegion: AT-9

Q1.4 WHO.IS?

1. contact for inria.fr
2. contact for fabien.info
3. contact for lemonde.fr

contact for inria.fr : Florian DUFOUR
contact for fabien.info : REDACTED FOR PRIVACY
contact for lemonde.fr : SOCIETE EDITRICE du monde

Q1.5 CURL (or WGET)

1. Ten first lines:
   curl -o Paris.html -L -H "Accept: text/html" http://dbpedia.org/resource/Paris
   curl -o Paris-rdf.xml -L -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Paris
2. Ten first lines for HTML and RDF http://ns.inria.fr/fabien.gandon#me
3. Ten first lines for HTML and RDF for ‘Vienna’ on Dbpedia
4. Ten first lines for the “URI of the name of Victor Hugo” in the Library of Congress:
   http://id.loc.gov/authorities/names/n79091479
5. Ten first lines for HTML and RDF
   https://purl.uniprot.org/uniprot/P43121
6. What is the topic and format of data obtained with
   curl -o data.json -L -H "Accept: application/json"
   https://www.wikidata.org/wiki/Special:EntityData/Q551861
7. What is the topic and format of data obtained with
   curl -o data.ttl -L -H "Accept: text/turtle" http://dx.doi.org/10.1007/3-540-45741-0_18

1.

- **curl -o Paris.html -L -H "Accept: text/html" http://dbpedia.org/resource/Paris** :

About: Paris

An Entity of Type: city, from Named Graph: http://dbpedia.org, within Data Space: dbpedia.org

Paris (French pronunciation: [paʁi]) is the capital and most populous city of France, with an estimated population of 2,165,423 residents in 2019 in an area of more than 105 km² (41 sq mi), making it the 30th most densely populated city in the world in 2020. Since the 17th century, Paris has been one of the world's major centres of finance, diplomacy,

- **curl -o Paris-rdf.xml -L -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Paris** :

<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#

2.

- **curl -o inria.html -L -H "Accept: text/html" http://ns.inria.fr/fabien.gandon#me:**

FOAF profile of Fabien GANDON
You may have been redirected here by your browser.
You can access Fabien GANDON's foaf profile in RDF or Fabien GANDON's homepage in HTML.

- **curl -o inria-rdf.xml -L -H "Accept: application/rdf+xml" http://ns.inria.fr/fabien.gandon#me:**

```
<?xml version='1.0' encoding='utf-8' ?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xml:base="http://ns.inria.fr/fabien.gandon">

  <foaf:PersonalProfileDocument rdf:about="">
    <foaf:maker rdf:resource="#me"/>
<foaf:primaryTopic rdf:resource="#me"/>
```

3.

HTML VERSION

About: Vienna

An Entity of Type: Capital city, from Named Graph: http://dbpedia.org, within Data Space: dbpedia.org
Vienna (/viˈɛnə/ vee-EN-ə; German: Wien [viːn]; Austro-Bavarian: Wean [veɐ̯n]) is the capital, largest city, and one of nine states of Austria. Vienna is Austria's most populous city and its primate city, with about two million inhabitants (2.9 million within the metropolitan area, nearly one third of the country's population), and its cultural, economic, and political center. It is the 6th-largest city proper by population in the European Union and the largest of all cities on the Danube river.

XML VERSION: curl -o htm_vienna-rdf.xml -L -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Vienna

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:skos="http://www.w3.org/2004/02/skos/core#"
        xmlns:dbp="http://dbpedia.org/property/"
        xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
        xmlns:dbo=http://dbpedia.org/ontology/
```

4.

XML VERSION

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:bf="http://id.loc.gov/ontologies/bibframe/"
xmlns:bflc="http://id.loc.gov/ontologies/bflc/" xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:cc="http://creativecommons.org/ns#" xmlns:foaf="http://xmlns.com/foaf/0.1/">
<madsrdf:PersonalName rdf:about="http://id.loc.gov/authorities/names/n79091479"
xmlns:madsrdf="http://www.loc.gov/mads/rdf/v1#">
<rdf:type rdf:resource="http://www.loc.gov/mads/rdf/v1#Authority"/>
<madsrdf:authoritativeLabel>Hugo, Victor, 1802-1885</madsrdf:authoritativeLabel>
<madsrdf:elementList rdf:parseType="Collection">

Html version

Hugo, Victor, 1802-1885

- URI(s)
  - http://id.loc.gov/authorities/names/n79091479
- Variants
  - Hiwkō, Vik‘t‘or, 1802-1885
  - Hīĵū, Fīktūr, 1802-1885
  - Giugo, Viktor, 1802-1885
  - Hsiao-o, 1802-1885
  - Hyowgo, Viktor, 1802-1885

5. The data obtained is in json format and the page is about Xavier Dolan
6. The data obtained is in turtle format and the page is about Distributed Artificial Intelligence for Distributed Corporate Knowledge Management written by fabien-gandon and Rose Dieng-Kuntz.

Q1.6 Find the URIs of «Pedro Almodóvar » on the Spanish Dbpedia and on Wikidata.

- **Spanish Dbpedia**: http://es.dbpedia.org/resource/Pedro_Almodóvar
- **Wikidata**: http://www.wikidata.org/entity/Q55171

Q1.7 Spotlight demo

Reproduce the demo:

1. Copy a text from Wikipedia (e.g. Muse Band page)
2. Find the DBpedia Spotlight service page
3. Paste the text and run the detection
4. Try with other texts and copy-paste one of the results you get.

Confidence: 0.5

Language: English

n-best candidates

SELECT TYPES... ANNOTATE

Muse are an English rock band from Teignmouth, Devon, formed in 1994. The band consists of Matt Bellamy (lead vocals, guitar, keyboards), Chris Wolstenholme (bass guitar, backing vocals), and Dominic Howard (drums, percussion).
Muse released their debut album, Showbiz, in 1999, showcasing Bellamy's falsetto and a melancholic alternative rock style. Their second album, Origin of Symmetry (2001), incorporated wider instrumentation and romantic classical influences and earned them a reputation for energetic live performances.[1] Absolution (2003) saw further classical influence, with strings on tracks such as "Butterflies and Hurricanes", and was the first of seven consecutive UK number-one albums.
Black Holes and Revelations (2006) incorporated electronic and pop elements, displayed in singles such as "Supermassive Black Hole",[1] and brought Muse wider international success. The Resistance (2009) and The 2nd Law (2012) explored themes of government oppression and civil uprising and cemented Muse as one of the world's major stadium acts. Topping the US Billboard 200, their seventh album, Drones (2015), was a concept album about drone warfare and returned to a harder rock sound. Their eighth album, Simulation Theory (2018), prominently featured synthesisers and was influenced by science fiction and the simulation hypothesis. Their ninth album, Will of the People (2022), which combined many genres and themes from their previous albums, was released in August 2022.
Muse have won numerous awards, including two Grammy Awards, two Brit Awards, five MTV Europe Music Awards and eight NME Awards. In 2012, they received the Ivor Novello Award for International Achievement from the British Academy of Songwriters, Composers and Authors. As of June 2016, they had sold more than 30 million albums worldwide.[2]

BACK TO TEXT

This demo uses the statistical DBpedia-Spotlight web service at https://api.dbpedia-spotlight.org/en.

How to cite this work

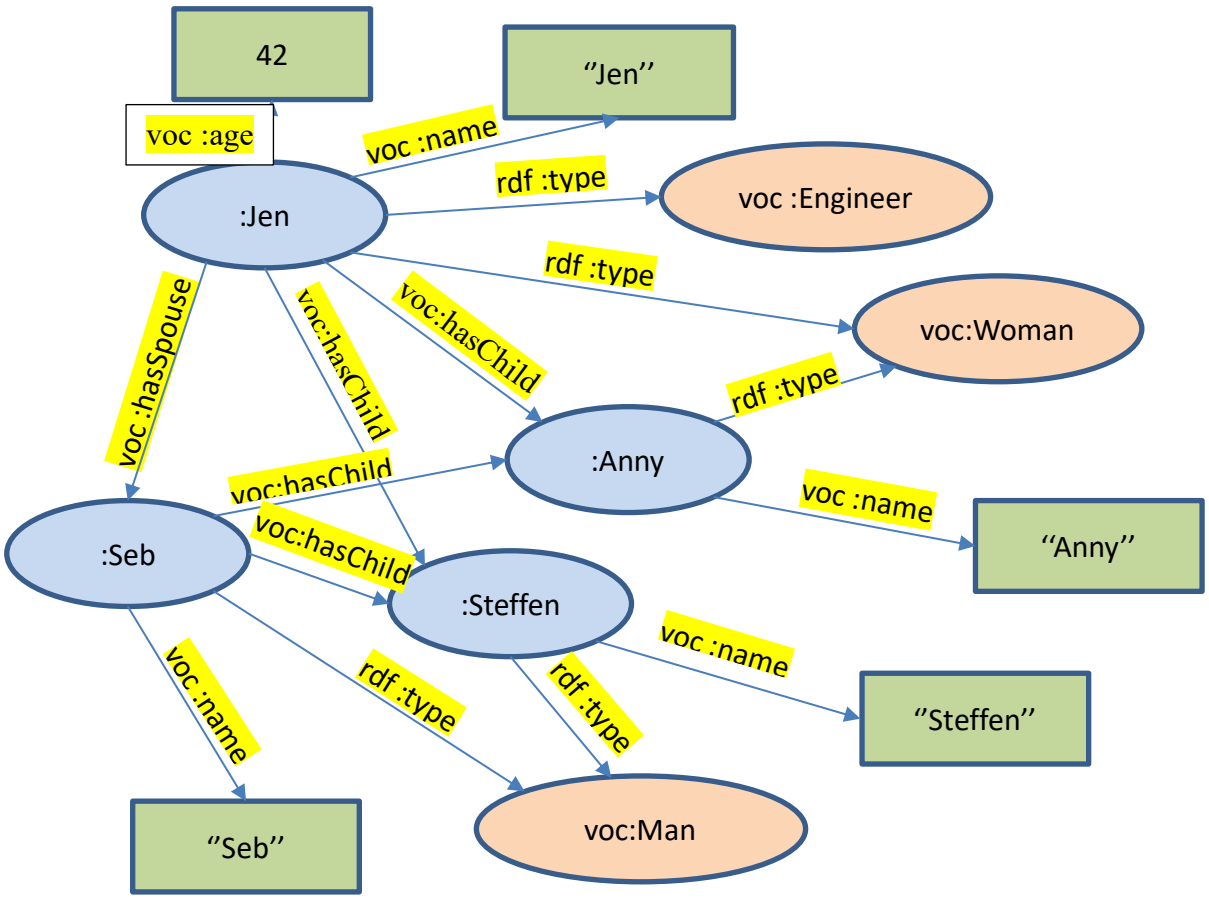
Questions from the course on RDF.

Q2.0 Fill the blanks.

"Jen is an engineer woman, 42-year-old, married to Seb who is a man with whom she had two children: Anny who is a woman and Steffen who is a man". For each person we also explicitly specify the name.

To fill the blanks we use the values: `:Seb, :Steffen, voc:name, voc:hasChild, voc:age, voc:hasSpouse, rdf:type, voc:Engineer, voc:Man, "Jen", "Seb", "Anny", "Steffen"`

For each person we also explicitly specify the name



Q2.1 What is missing to say that "doc.html has for authors Catherine and Fabien and is about Music and Piano" ?

```
<http://inria.fr/rr/doc.html> <http://inria.fr/schema#author>
  <http://ns.inria.fr/fabien.gandon#me> .
<http://inria.fr/rr/doc.html> <http://inria.fr/schema#author>
"Catherine" .

<http://inria.fr/rr/doc.html> <http://inria.fr/schema#theme> "Music" .
<http://inria.fr/rr/doc.html> <http://inria.fr/schema#theme> "Piano" .
```

Q2.2 Fill the blanks (N3/Turtle)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://www.unice.fr/voc#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.unice.fr/data#Jen> a voc:Engineer , voc:Woman ;
    voc:age "42"^^xsd:string ;
    voc:hasChild
<http://www.unice.fr/data#Anny>,<http://www.unice.fr/data#, Steffen>;
    voc:hasSpouse <http://www.unice.fr/data#Seb> ;
    voc:name "Jen" .
<http://www.unice.fr/data#Seb> a voc:Man ;
    voc:hasChild <http://www.unice.fr/data#Anny>,
        <http://www.unice.fr/data#Steffen> ;
    voc:name "Seb" .
<http://www.unice.fr/data#Anny> a voc:Woman ;
    voc:name "Anny" .
< http://www.unice.fr/data#Steffen> a voc:Man ;
     Voc:name “Steffen” .
```

Q2.3 Fill the blanks (RDF/XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [   <!ENTITY vocab "http://www.unice.fr/voc">
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"> ]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:voc="&vocab;#" xml:base="http://www.unice.fr/data">
 <voc:Woman rdf:about="#Jen">
    <voc:name>Jen</voc:name>
    <voc:age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42
</voc:age>
    <voc:hasSpouse rdf:resource="#Seb"></voc:hasSpouse>
    <voc:hasChild rdf:resource="#Steffen"></voc:hasChild>
    <voc:hasChild>
      <rdf:Description rdf:about="#Anny">
        <voc:name>Anny</voc:name>
        <rdf:type rdf:ressource="&vocab;#Woman"></rdf:type>
      </rdf:Description>
    </voc:hasChild>
    <rdf:type rdf:resource="&vocab;#Engineer"></rdf:type >
 </voc:Woman>
 <voc:Man rdf:about="#Seb">
    <voc:name>Seb</voc:name>
    <voc:hasChild rdf:resource="#Steffen"></voc:hasChild>
    <voc:hasChild rdf:resource="#Anny"></voc:hasChild>
  </voc:Man>
  <voc:Man rdf:about="#Steffen">
    <voc:name>Steffen</voc:name>
  </voc:Man>
</rdf:RDF>
```

Q2.4 Visit me please

Get the RDF data from: http://ns.inria.fr/fabien.gandon#me

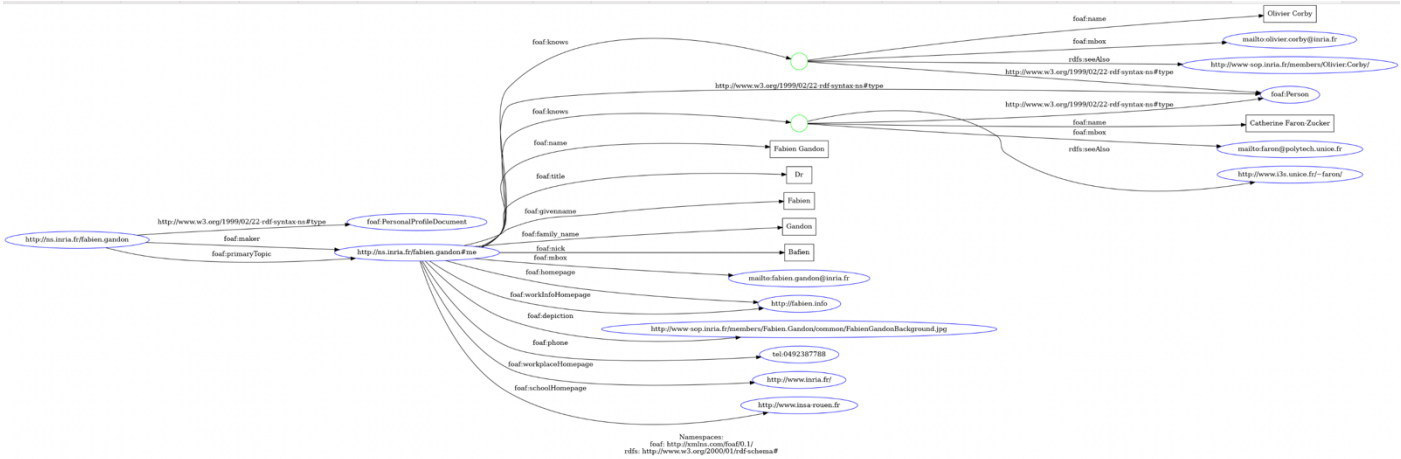
1. Get the RDF data from: http://ns.inria.fr/fabien.gandon#me
2. What is the syntax used?
3. Validate it and see the graph (RDF/XML):
   http://www.w3.org/RDF/Validator/
4. Translate into Turtle/N3:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdf-translator.appspot.com/
5. Visualize it also with:
   http://cltl.nl/visualrdf/
   http://www.easyrdf.org/converter (PNG, SVG)
   https://www.ldf.fi/service/rdf-grapher
6. Adapt to your data and do it again

2. The syntax used is XML

4.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://ns.inria.fr/fabien.gandon>
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://ns.inria.fr/fabien.gandon#me> ;
  foaf:primaryTopic <http://ns.inria.fr/fabien.gandon#me> .

<http://ns.inria.fr/fabien.gandon#me>
  a foaf:Person ;
  foaf:name "Fabien Gandon" ;
  foaf:title "Dr" ;
  foaf:givenname "Fabien" ;
  foaf:family_name "Gandon" ;
  foaf:nick "Bafien" ;
  foaf:mbox <mailto:fabien.gandon@inria.fr> ;
  foaf:homepage <http://fabien.info> ;
  foaf:depiction <http://www-
sop.inria.fr/members/Fabien.Gandon/common/FabienGandonBackground.jpg> ;
  foaf:phone <tel:0492387788> ;
  foaf:workplaceHomepage <http://www.inria.fr/> ;
  foaf:workInfoHomepage <http://fabien.info> ;
  foaf:schoolHomepage <http://www.insa-rouen.fr> ;
  foaf:knows [
    a foaf:Person ;
    foaf:name "Olivier Corby" ;
    foaf:mbox <mailto:olivier.corby@inria.fr> ;
    rdfs:seeAlso <http://www-sop.inria.fr/members/Olivier.Corby/>
  ], [
    a foaf:Person ;
    foaf:name "Catherine Faron-Zucker" ;
    foaf:mbox <mailto:faron@polytech.unice.fr> ;
    rdfs:seeAlso <http://www.i3s.unice.fr/~faron/>
  ] .
```

5.



Q2.5 what is the meaning of this RDF? What is this description saying?

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:exs="http://example.org/schema#">
 <rdf:Description rdf:about="http://example.org/doc.html">
  <rdf:type rdf:resource="http://example.org/schema#Report"/>
  <exs:theme rdf:resource="http://example.org#Music"/>
  <exs:theme rdf:resource="http://example.org#Danse"/>
  <exs:nbPages
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">73</exs:nbPages>
 </rdf:Description>
</rdf:RDF>
```

The description says that “doc.htm” is of type “report”, has for theme “Music” and “Danse” and has “73” pages.

Q2.6 Visit to Victor Hugo

1. See HTML data from:
   http://id.loc.gov/authorities/names/n79091479.html
2. Get RDF data from:
   http://id.loc.gov/authorities/names/n79091479.rdf
3. What is the syntax?
4. Translate into Turtle/N3:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/
5. Any remark about the values of the properties of Victor Hugo?

3. The syntax used is RDF/XML

4.

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix ns0: <http://www.loc.gov/mads/rdf/v1#> .
@prefix ns1: <http://id.loc.gov/ontologies/bflc/> .
@prefix ns2: <http://id.loc.gov/vocabulary/identifiers/> .
@prefix ns3: <http://id.loc.gov/ontologies/RecordInfo#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix skosxl: <http://www.w3.org/2008/05/skos-xl#> .
@prefix ns4: <http://purl.org/vocab/changeset/schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ns5: <http://id.loc.gov/datatypes/edtf/> .
```

```
<http://id.loc.gov/authorities/names/n79091479>
  a <http://www.loc.gov/mads/rdf/v1#PersonalName>,
<http://www.loc.gov/mads/rdf/v1#Authority>, skos:Concept ;
  ns0:authoritativeLabel "Hugo, Victor, 1802-1885" ;
  ns0:elementList (
    _:genid1
    _:genid3
  ) ;
  ns0:hasVariant [
    a ns0:PersonalName, ns0:Variant ;
    ns0:variantLabel "Hiwkō, Vik‘t‘or, 1802-1885" ;
    ns0:elementList (
      _:genid6
      _:genid8
    )
  ]
```

6. We can see that for the property ns0: variantLabel, the name of Victor Hugo has been given in different languages but they did not precise which one which is damageable.

Q2.7 What is the syntax of the following RDF statement? What does it mean?

```
@prefix dcterms: <http://purl.org/dc/terms/>.
GRAPH  <http://inria.fr/data/algebra>
{
  <http://inria.fr/rr/doc.html>
    dcterms:subject
      <http://data.bnf.fr/ark:/12148/cb121105993> .
}
```

The syntax used is TriG. It means:
In the name graph “http://inria.fr/data/algebra”, we are putting all the triples that are inside the curly brace. So if we need to talk about the triple contain inside the curly brace we only have to specify “http://inria.fr/data/algebra” as our subject.

Q2.8 Visit Leukocyte surface antigen CD53

1. See HTML data from:
   http://www.uniprot.org/uniprot/Q61451
2. Get RDF data from:
   http://www.uniprot.org/uniprot/Q61451.rdf
3. What is the syntax?
4. Translate into Turtle/N3:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/
5. Any remark about the structure of the data?

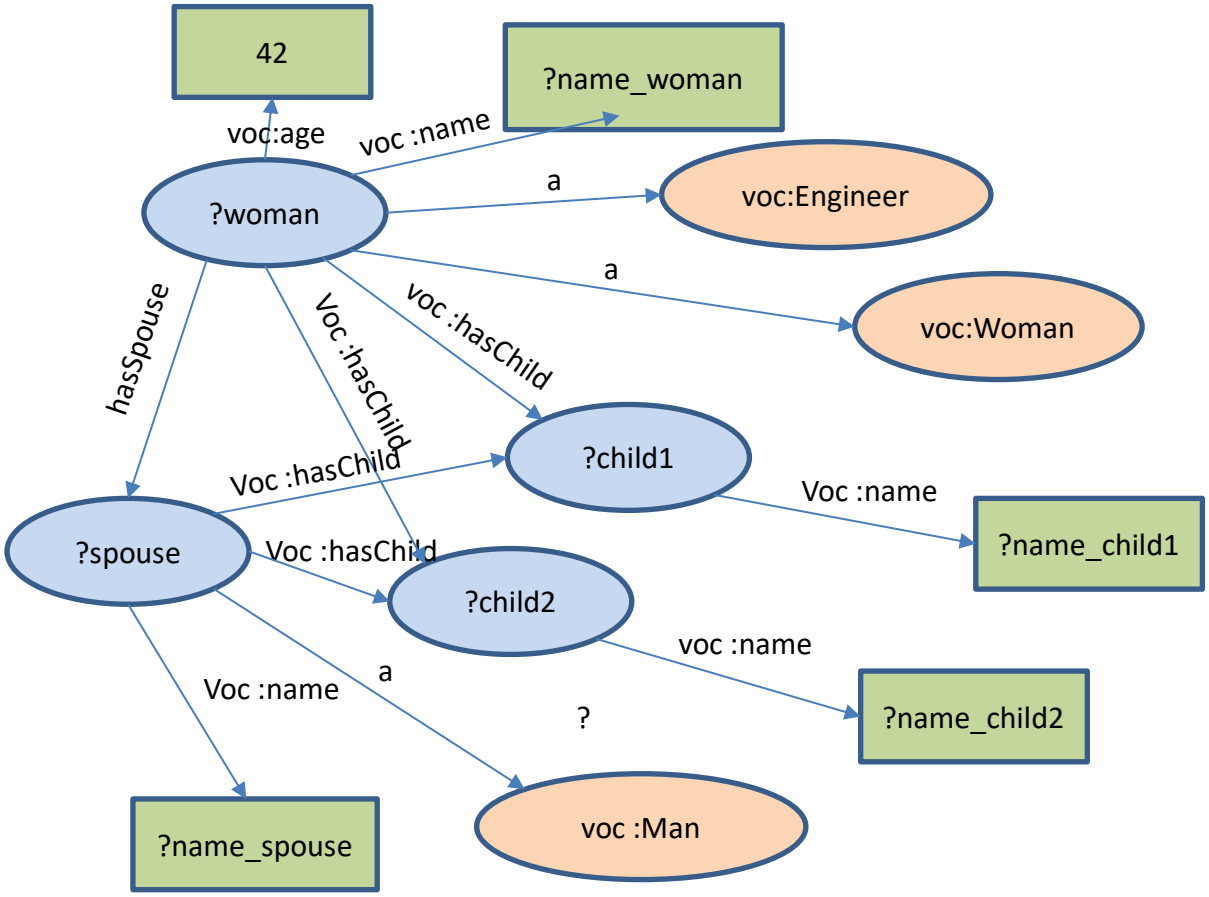
2. the synthax is RDFXML
5. They use reification in their code. The old mechanism of RDF to talk about triple.

Questions from the course on SPARQL.

Q2.0 Fill the blanks

"find women, 42-year old, married to Seb who is a man with whom they had two children and get the name of every one."

To fill the blanks we use the values: `?woman, ?child1, ?spouse, ?child2, voc:name, voc:hasChild, voc:age, voc:hasSpouse, a, voc:Engineer, voc:Man, ?name_woman, ?name_spouse, ?name_child1, ?name_child2`



Q3.1 Test SPARQL online

Connect to: https://corese.inria.fr/srv/tutorial/sparql

Answers to the query:

```
prefix v: <http://www.inria.fr/2015/humans#>
select * where {  ?x a v:Person . }
```

1 <http://www.inria.fr/2015/humans-instances#John>
2 <http://www.inria.fr/2015/humans-instances#Sophie>
3 <http://www.inria.fr/2015/humans-instances#Mark>
4 <http://www.inria.fr/2015/humans-instances#Eve>

5 <http://www.inria.fr/2015/humans-instances#David>
6 <http://www.inria.fr/2015/humans-instances#Laura>
7 <http://www.inria.fr/2015/humans-instances#William>
8 <http://www.inria.fr/2015/humans-instances#Karl>

Q3.2 Test SPARQL online

Connect to

http://dbpedia.org/snorql/ or

http://fr.dbpedia.org/sparql or …

http://wiki.dbpedia.org/Internationalization/Chapters

Answers to the query:

```
SELECT ?x ?p ?v WHERE {
  ?x rdfs:label "Paris"@fr .
  ?x ?p ?v .
}
LIMIT 10
```

x	p	v
:Paris	-	owl:Thing
:Paris	-	dbpedia:ontology/Place
:Paris	-	dbpedia:ontology/Location
:Paris	-	<http://schema.org/Place>
:Paris	-	<http://www.wikidata.org/entity/Q486972>
:Paris	-	dbpedia:class/yago/WikicatArchaeologicalSitesInFrance
:Paris	-	dbpedia:ontology/PopulatedPlace
:Paris	-	dbpedia:class/yago/WikicatCapitals
:Paris	-	dbpedia:class/yago/WikicatCapitalsInEurope
:Paris	-	dbpedia:class/yago/WikicatCitiesInFrance

Q3.3 Test SPARQL online

Connect to:

https://query.wikidata.org/

What does this query retrieve?

```
SELECT distinct ?p ?n WHERE
{     wd:Q30 p:P6 [ ps:P6 ?p ].
      ?p rdfs:label ?n .
      FILTER (lang(?n)="en") }
```

Discover wd:Q30 using the namespace attached to wd:

PREFIX wd: <http://www.wikidata.org/entity/>

Discover p:P6 using the namespace attached to p:

PREFIX p: <http://www.wikidata.org/prop/>

PREFIX ps: <http://www.wikidata.org/prop/statement/>

Find q-name of the property “given name”

https://www.wikidata.org/wiki/Wikidata:List_of_properties

The query retrieves the name of the different president of the United States

The q-name of the property “given name” is P735

Q3.4 SPARQL query to return 20 persons at most (use type foaf:Person)

```
SELECT *
WHERE {?x a foaf:Person}
LIMIT 20
```

Q3.5 SPARQL query to return 20 persons (at most), after the 10th result i.e. from 11th to 30th

```
SELECT  *
WHERE {?x a foaf:Person}
LIMIT 20
OFFSET 10
```

Q3.6 You have two properties: c:name and c:age

1.Find the age of resources whose name is 'Fabien'

```
SELECT ?age
WHERE {
 ?x c:name ?name ;  c:age ?age
}
```

2.Find the name of resources whose age is less than 50

```
SELECT ?name
WHERE { ?x c:name ?name;
 c:age ?age .
FILTER(?age<50)}
```

3. Find properties and their values for resources whose name is 'Fabien' and whose age is less than 50

```
SELECT ?p ?v
Where {
?x c:name "Fabien";
c:age ?age;
?p ?v .
FILTER(?Age<50)}
```

4.Find other names of resources with name 'Fabien', at least, a second name

```
select ?name
where { ?x :name 'Fabien', ?name .
 filter (?name != 'Fabien')}
```

5.Find resources which have two different properties with the same value

```
SELECT ?x
WHERE{
?x ?p ?v; ?q ?v .
FILTER(?p!=?q)
}
```

6.Find resources which have the same property with two different values

```
SELECT ?x
WHERE {
?x ?p ?v1; ?p ?v2 .
FILTER(?v1 != ?v2)}
```

Q3.7 Could this query return `ex:a c:memberOf ex:b` and why ?

```
select * where {
  ?x c:memberOf ?org .
  minus { ex:a c:memberOf ex:b }
}
```

No it won't return return `ex:a c:memberOf ex:b` because there is ?x is not reuse in the minus part, so the system does not know what to do with that part.

Q3.8 get the members of organizations (c:memberOf) but remove the resources author of a document (c:author) by using 'not exists'

SELECT *
WHERE {
?x c:memberOf ?org .
FILTER(not exist {?y c:author ?doc}
}

Q3.9 what is retrieving this query ?

```
prefix ex: <http://example.org/>
select ?x (count(?doc) as ?c)
where {  ?x ex:author ?doc  }
group by ?x
order by desc(count(?doc))
```

This query looks for an author who wrote a document, it group by the different resources find and order them in descending number of written document. It return the number of document and stored the result in the variable c.

Q3.10 What expression should we use to find the ?x related to ?y by paths composed of properties foaf:knows and/or rdfs: seeAlso?

- ?x (foaf:knows | rdfs:seeAlso)+ ?y
- ?x foaf:knows+ | rdfs:seeAlso+ ?y
- ?x (foaf:knows / rdfs:seeAlso)+ ?y

We should use the first expression • ?x (foaf:knows | rdfs:seeAlso)+ ?y because it allows all the combination using foaf:knows and rdfs:seeAlso to find the value ?y

Q3.11 what is this query retrieving?

```
prefix foaf: <http://xmlns.com/foaf/0.1/>
select ?x (if (bound(?n), ?n, "John Doe") as ?m)
where {
   ?x foaf:knows ?y
   optional { ?y foaf:name ?n }
}
```

We are looking for a variable that knows a person ?y. ?y can optionally have a name. We select the resources found ?x and if we have found a name for ?y we stored ?n in ?m otherwise we store "John Doe" in ?m

Q3.12 what is this query retrieving?

```
prefix ex:  <http://example.org/>
select ?x (avg(?a) as ?b)
where {
  ?x ex:knows ?y .
  ?y ex:age ?a
}
group by ?x
```

We are looking for a resource ?x that knows ?y. ?y must have an age ?a. We group by resource found ?x. We return the resource ?x and the variable ?b which is the average of the age found (?a)

Q3.13 You have two properties: c:name and c:study and the resources c:Informatics and c:Mathematics

1. Find resources that study informatics or mathematics
2. In addition return the name of the resource if it has a name
3. In addition return the graph where the name is given

1.

```
Select *
Where { {?x c:study c:informatics }
UNION
{?x c:study c:Mathematics }}
```

2.

```
Select *
Where {
{?x c:study c:informatics }
UNION
{?x c:study c:Mathematics }
OPTIONAL{?x c:name ?name}
}
```

3.

```
Select *
Where {
{?x c:study c:informatics }
UNION
{?x c:study c:Mathematics }
OPTIONAL{GRAPH ?graph_name {?x c:name ?name}}
}
```

Q3.14 On which graph(s) is calculated ?x ?p ?y

On which graph(s) is calculated graph ?g { ?y ?q ?z }

```
prefix ex:  <http://example.org/>
select *
from ex:g1
from named ex:g2
where {
   ?x ?p ?y .
   graph ?g { ?y ?q ?z}  }
```

<Skip for lack of time>

Q3.15 Write a query to change foaf:name into rdfs:label

DELETE{?x foaf:name ?y}
INSERT {?x rdfs:label ?y}
WHERE{?x foaf:name ?y}

Q3.16 what is this query performing?

```
prefix ex: <http://example.org/>
delete { ?x ex:age ?a }
insert { ?x ex:age ?i }
where  {
  select ?x ?a (xsd:integer(?a) as ?i)
  where {
    ?x ex:age ?a
    filter(datatype(?a) = xsd:string)
  }
}
```

This query transforms age values from strings to integers for resources where age values are currently stored as strings. It replaces the old age value (?a) with the new integer value (?i).

Q3.17 Which clauses could you use to obtained results as RDF triples following a specific pattern?

- SELECT … WHERE {…} …
- CONSTRUCT { } WHERE {…} …
- DESCRIBE <…> DESCRIBE … {…}
- ASK {…}
- DELETE { … } INSERT { … } WHERE {…} …

We can use CONSTRUCT { } WHERE {…} …

Q3.18 What is the difference between these two queries?

```
prefix ex: <http://example.org/>
Insert { ?x a ex:Parent }
where { ?x ex:hasChild  ?y }


prefix ex: <http://example.org/>
construct { ?x a ex:Parent }
where  { ?x ex:hasChild  ?y }
```

In the first query we look for resource that has the property hasChild and then we insert triple to say that the resource is a parent.

In the second query, we look for the same thing but we have the possibility to migrate the data from a database to another, change from one schema to another from one structured to another

---

Questions from the course on Ontologies.

Q4.0 Choose among the following assertions one or more you consider to be true:

- ☐ an ontology is necessarily formalized in first-order logic
- ☐ an ontology may allow inferences on data that uses it
- ☐ conceptual graphs can represent an ontology
- ☐ a shared ontology promotes interoperability
- ☐ description logics can represent an ontology

an ontology may allow inferences on data that uses it.
conceptual graphs can represent an ontology.
a shared ontology promotes interoperability.
description logics can represent an ontology

Q4.1 work alone for 10 minutes

```
From real example of ontology engineering: you are designing a
system to assist (data) management of species presented in a museum
of natural sciences. In particular you need to organize at least
the species below. Structure them and add categories as needed.

Species: animal, bird, cat, cicada, clam, crocodile, dog,
dragonfly, fish, frog, fungus, insect, kiwi, octopus, ostrich,
shark, snail, snake, spider, thing, trout, whale
```

1. quatic animals:
    - Fish
    - Octopus
    - Shark
    - Whale
    - Trout
    - Clam
    - animal
2. Terrestrial animals:
    - Cat
    - Cicada
    - Crocodile
    - Dog
    - Frog
    - Kiwi
    - Ostrich
    - Snail
    - Snake
    - Spider
    - insect
    - Bird
    - Dragonfly

- Insect
- Thing
- Animal
- Fungus

Questions from the course on RDFS

Q4.2 RDFS contains primitives to (several answers possible)…

- ☐ describe classes of resources
- ☐ describe formulas of calculation for values of properties
- ☐ describe types of properties of resources
- ☐ document definitions in natural language
- ☐ sign and authenticate the authors of the definitions of classes and properties

describe classes of resources
describe types of properties of resources
document definitions in natural language (rdfs label)

Q4.3. What is defined and derived from these definitions?

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@prefix : <http://inria.fr/devices#>
 :Phone rdfs:subClassOf  :Device .
 :Computer rdfs:subClassOf  :Device .
 :Smartphone rdfs:subClassOf :Computer .
 :Smartphone rdfs:subClassOf :Phone .
```

We can say that Smartphone is a subclass of Device

Q4.4. What is defined and derived from these definitions?

```
@prefix rdfs:  < http://www.w3.org/2000/01/rdf-schema# >
@prefix : <http://inria.fr/member#>
:employeeOf rdfs:subPropertyOf  :proRelationWith .
:hasControlOver rdfs:subPropertyOf :proRelationWith .
:isShareholderOf rdfs:subPropertyOf :hasControlOver .
:isCEOof rdfs:subPropertyOf :employeeOf, :hasControlOver .
```

The following relation will be added in the system :
:isCEOof rdfs:subPropertyOf :proRelationWith .
:isShareholderOf rdfs:subPropertyOf :proRelationWith .

Q4.5. Download the ontology Schema.org founded by Google, Microsoft, Yahoo and Yandex:
https://schema.oschrg/version/latest/schemaorg-current-https.ttl
Find the Class schema:AboutPage and identify its super-class
Can a document be of type schema:abstract ?

It's super-class is schema:WebPage
schema:abstract is not a class but a property and this is not okay to say that a document is a property. So a document cannot be of type schema:abstract.

Q4.6. What can be said about the types of the resources that will be linked by the properties defined below?

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix : <http://inria.fr/humans#>
:driverOf  rdfs:subPropertyOf  :isControling .
:piloteOf  rdfs:subPropertyOf  :isControling .
:isControling  rdfs:domain  :Human ;  rdfs:range :Object .
:driverOf rdfs:range  :Car .
:piloteOf rdfs:domain  :Adult ; rdfs:range  :Plane .
```

For example:
?x :piloteOf ?y => the system will infer that ?x is an adult and given that :piloteOf is a subproperty of isControling, the system will also infer that ?x is a Human. By applying the same reasoning, the system will add triples to define ?y as a Plane and another one to define it as an object
?x :driverOf ?y. => ?y is a car and by extension an object. ?x is a human

Q4.7. What could we add to this schema (several answers are possible)?

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@ prefix :   <http://ns.inria.fr/humans/schema#>
:p1  a rdf:Property ;  rdfs:label  "age"@en .
:c1  a rdfs:Class ; rdfs:comment "a human being"@en .
```

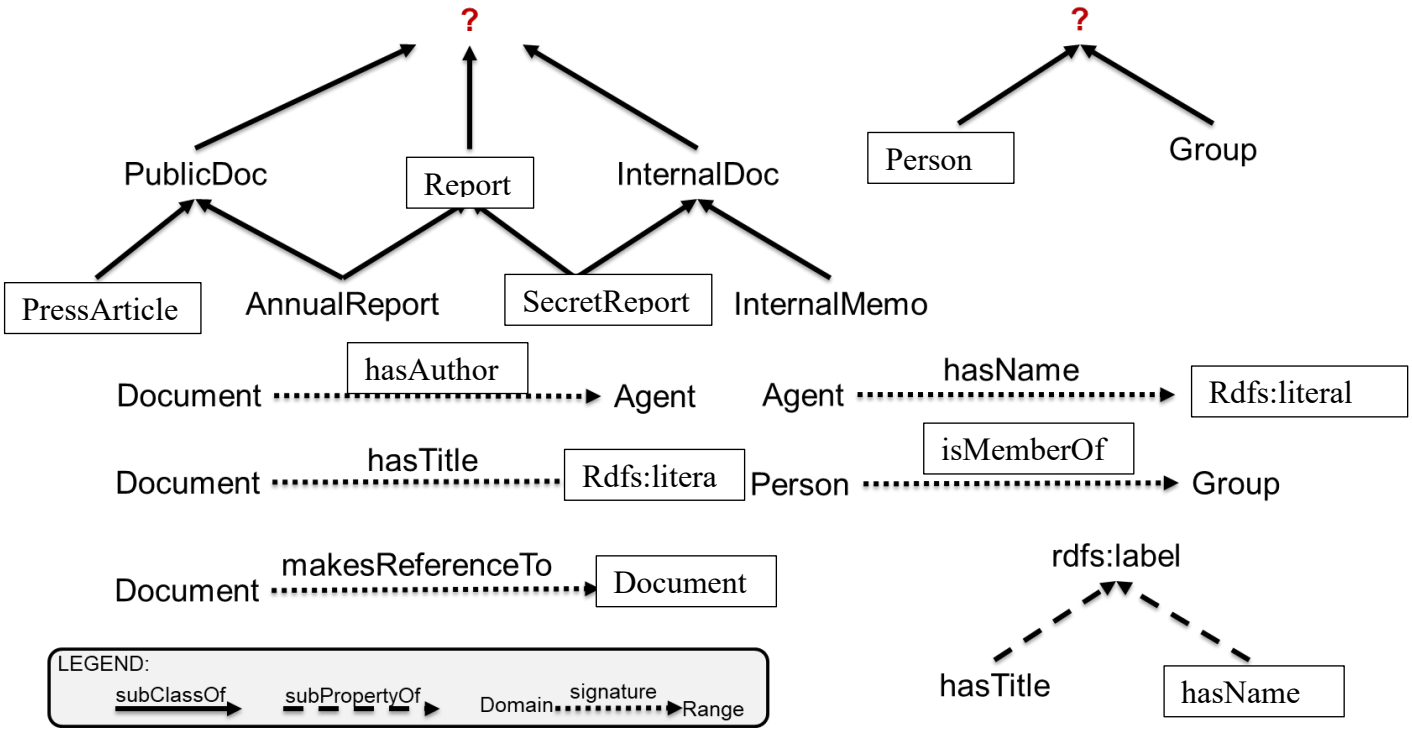
- ☐ :p1 rdfs:label "firstname"@en, "prénom"@fr .
- ☐ :c1 rdfs:comment "un être humain"@en .
- ☐ :c1 rdfs:label "person"@en, "personne"@fr .
- ☐ :p1 rdfs:label "âge"@fr .
- ☐ :c1 rdfs:label "woman"@en .
- ☐ :c1 rdfs:label "persona"@es .
- ☐ :p1 rdfs:comment "the length of time something has existed."@en .

:c1 rdfs:label "person"@en, "personne"@fr .
:p1 rdfs:label "âge"@fr .
:c1 rdfs:label "persona"@es .
:p1 rdfs:comment "the length of time something has existed."@en .

Q4.8. (a) Fill the blanks with: `Document, PublicDoc, PressArticle, Report, AnnualReport, InternalDoc, SecretReport, InternalMemo, Agent, Person, Group, hasTitle, hasAuthor, makesReferenceTo, hasName, isMemberOf` + rdf / rdfs primitives.
(b) Write it in RDFS and validate the RDF.

Documents

Agen


?
?
PublicDoc
Report
InternalDoc
Person
Group
PressArticle
AnnualReport
SecretReport
InternalMemo
Document
hasAuthor
Agent
Agent
hasName
Rdfs:literal
Document
hasTitle
Rdfs:litera
Person
isMemberOf
Group
Document
makesReferenceTo
Document
rdfs:label
hasTitle
hasName
LEGEND:
subClassOf
subPropertyOf
Domain
signature
Range

Questions from the course on OWL.

Q5.1 What is asserted and what can we deduce?

```
ex:Man owl:intersectionOf (ex:Male ex:Human) .
ex:Woman owl:intersectionOf (ex:Female ex:Human) .
ex:Human owl:unionOf (ex:Man ex:Woman) .
ex:Jane a ex:Human .
ex:John a ex:Man .
ex:James a ex:Male .
ex:Jane a ex:Female .
```

We are asserting that,
a man is the intersection of a male and a human.
woman is the intersection of a female and a human.
Human is the union of a man and a woman.

ex:Jane a ex:Human .
We can just say that jane is a human, we cannot infer anything.
ex:John a ex:Man .
John is a man, we can infer that he is also a Male and a Human
ex:James a ex:Male .
we can just say that James is a male and we cannot infer anything.
ex:Jane a ex:Female .
with that extra information, we know that Jane is a Human and a Female, so she is a woman.

Q5.2 What are we defining and inferring?

```
@prefix ex: <http://example.org/>

ex:GrandFather rdfs:subClassOf [
   a owl:Class ;
   owl:intersectionOf ( ex:Parent ex:Man )
] .

ex:Jim a ex:Man, ex:Parent .
ex:Jack a ex:GrandFather .
```

GrandFather is a subclassof a owl class which is the intersection of parent and man.

- Jim is a man and a parent, we cannot infer that Jim is a GrandFather because he can be a parent and a man without being a grandfather.
- Jack is a grandfather, so he is a parent and a man.

Q5.3 What is asserted and what can we deduce?

```
ex:hasSpouse a owl:SymmetricProperty .
ex:hasChild  owl:inverseOf ex:hasParent .
ex:hasParent rdfs:subPropertyOf ex:hasAncestor .
ex:hasAncestor a owl:TransitiveProperty .
ex:Jim  ex:hasChild  ex:Jane .
ex:Jane ex:hasSpouse ex:John .
ex:Jim  ex:hasParent ex:James .
```

The property hasSpouse is sysmetric.
HasChild is the inverse of hasParent.
hasParent is a subproperty of hasAncestor.

has Ancestor is a transitive property.

ex:Jim ex:hasChild ex:Jane . ➔ (Jane ex :hasParent Jim .), (Jane ex :hasAncestor Jim) (Jane ex:hasAncestor James)
ex:Jane ex:hasSpouse ex:John . ➔ ex :John : hasSpouse Jane
ex:Jim ex:hasParent ex:James . ➔ (Jim ex :hasAncestor ex :James), (ex:James ex:hasChild ex:Jim .)

Q5.4 What is asserted and what can we deduce?

```
ex:Human owl:equivalentClass foaf:Person .
foaf:name owl:equivalentProperty ex:name .
ex:JimmyPage a ex:Human ;
             owl:sameAs ex:JamesPatrickPage .
ex:JimmyHendrix owl:differentFrom ex:JimmyPage .
```

Assertion :
Ex:Human is a class equivalent to foaf:Person
Foaf:name is an equivalent property to ex:name
JimmyPage is a Human, and it is the same as ex:JamesPatrickPage,
JimmyHendrix is not the same as JimmyPage

We can infer that:
:JimmyPage is a foaf:Person so ex:JamesPatrickPage is a human and a foaf:Person as he is the same as :JimmyPage.

Q5.5 What are we defining and inferring?

```
ex:UnhappyPerson owl:equivalentClass [
  a owl:Class ;
  owl:intersectionOf (
    ex:Person
    [ a owl:Class ; owl:complementOf ex:Happy ]
  )
] .
```

Ex:UnhappyPerson is a class which is equivalent to an anonymous class which is the intersection of ex:Person and a another anonymous class which is the complement of ex:Happy.
So we are inferring that in the class of Person we will find 2 types of resources : the Unhappy Person and the Happy Person because they are complementary in the class of Person.

Q5.6 What is asserted and what can we deduce?

```
ex:Human rdfs:subClassOf
  [ a owl:Restriction ;
    owl:onProperty ex:hasParent ;
    owl:allValuesFrom ex:Human ] .
ex:Tom a ex:Human .
ex:Tom ex:hasParent ex:James, ex:Jane.
```

Ex:Human is a subsclass of an anonymous class which is the restriction on hasParent. hasParent must take all its value in the set ex:Human.
Tom is a ex:Human so when we xill use the property hasParent , the object must be human so :
From **ex:Tom ex:hasParent ex:James, ex:Jane.** We can infer that James ex:James and ex:Jane are human

Q5.7 What are we defining and inferring?

```
@prefix ex: <http://example.org/>
ex:PersonList rdfs:subClassOf
  [
    a owl:Restriction ;
    owl:onProperty rdf:first ;
    owl:allValuesFrom ex:Person
  ] ,  [
    a owl:Restriction ;
    owl:onProperty rdf:rest ;
    owl:allValuesFrom ex:PersonList
  ] .

ex:value rdfs:range ex:PersonList .
ex:abc ex:value (ex:a ex:b ex:c) .
```

ex :PersonList is a subclass of two anonymous classes. The first one has a restriction on property who says that the first element of the list must take its values in ex:person.
The second class is also a restriction who says that the rest of the element of the list must takes all its values in ex:PersonList.
Ex:value takes its values in ex:PersonList .
ex:abc ex:value (ex:a ex:b ex:c) .
So from this triple , ex:a will be a ex:Person and the rest of the list (ex:b ex::c) will become a ex:PersonList and recursively ex:b will become a ex:person and ex:c will become a ex:PersonList and to finish ex:c will become a ex:Person.

So recursively every elements on the list will become ex:Person

Q5.8 What are we defining and inferring?

```
@prefix ex: <http://example.org/>
ex:Human rdfs:subClassOf [
  owl:intersectionOf (
    [
      a owl:Restriction ;
        owl:onProperty ex:hasBiologicalFather ;
        owl:maxCardinality 1
    ] , [
      a owl:Restriction ;
        owl:onProperty ex:hasBiologicalMother ;
        owl:maxCardinality 1
    ] )
] .
ex:Jane a ex:Human ;
        ex:hasBiologicalFather ex:James , ex:Jhon .
```

ex:Human is a subclass of the intersection of two anonymous classes. The first one is the restriction on the property hasBiologicalFather which can have a cardinality of 0 or 1.
The second classes is the restriction on the property hasBiologicalMother and which can have a cardinality of 0 or 1.
ex:Jane a ex:Human ; ex:hasBiologicalFather ex:James , ex:Jhon .
Jane is a human and has Two BiologicalFather James and Jhon.

From that triple the system will infer that James and Jhon are the same person. It will return an error if in the system James and Jhon are defined as two different person.

Q5.9 Visit the financial Industry Business Ontology (FIBO)
https://spec.edmcouncil.org/fibo/
with a lot of companies using it: https://edmcouncil.org/page/listofmembersreview

Negociate the turtle version of the part about governments:
https://spec.edmcouncil.org/fibo/ontology/BE/GovernmentEntities/GovernmentEntities/

Explain the formal definitions of Government, NationalGovernment, FederalGovernment, and isJurisdictionOf.

Questions from the course on Vocabularies.

Q6.0 Visit schema.org and

1. Check the documentation of schema:Accommodation
   https://schema.org/Accommodation
   and identify the more specific types in the page
2. Download the current version in Turtle (.ttl):
   https://schema.org/docs/developers.html
   extract the Turtle definition of schema:Recipe
   and find the name of the class it inherits from

The class of schema: Recipe inherits from schema:HowTo

Q6.1 What do you think of the annotation?

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
<#B-A-Ba> a skos:Concept ;
  skos:prefLabel   "B.A.-BA"@en , "b.a.-ba"@en ;
  skos:altLabel    "B-A-BA"@en ,  "b-a-ba"@en ;
  skos:hiddenLabel "BABA"@en , "baba"@en .
```

We can only have one preferred label. Here we have two so skos will raise an error.

Q6.2 practice:

1. Using the site prefix.cc find back the namespace usually associated to the SKOS prefix
2. Access the URL of the namespace and find the RDF source file defining the SKOS vocabulary
3. Find the definition of the property `narrowMatch` and give all the relations it has with other properties

```
skos:narrowMatch
  rdfs:label "has narrower match"@en ;
  rdfs:isDefinedBy <http://www.w3.org/2004/02/skos/core> ;
  skos:definition "skos:narrowMatch is used to state a hierarchical mapping link
between two conceptual resources in different concept schemes."@en ;
  a owl:ObjectProperty, rdf:Property ;
  rdfs:subPropertyOf skos:mappingRelation, skos:narrower ;
  owl:inverseOf skos:broadMatch .
```

4. Skos:narrowMatch is a subproperty of mapping relation and narrower. It is also the inverse of broad match.

Q6.3 practice:

1. Find and open the source file of Dublin Core Terms:
   https://dublincore.org/schemas/rdfs/
   Look at the definition of the class `FileFormat` and find the class it inherits from.
2. Choose your preferred book on Amazon, Fnac, etc. and describe it in an RDF annotation using as many DC primitives as necessary.
3. Add the most restrictive CC license to your preferred book ; is this license appropriate?

1.

```
dcterms:FileFormat
    dcterms:issued "2008-01-14"^^<http://www.w3.org/2001/XMLSchema#date> ;
```

```
    a rdfs:Class ;
    rdfs:comment "A digital resource format."@en ;
    rdfs:isDefinedBy <http://purl.org/dc/terms/> ;
    rdfs:label "File Format"@en ;
    rdfs:subClassOf dcterms:MediaType .

Fileformat is a subclass or inherits from dcterms:MediaType.
```

Question 2 and 3:

@prefix cc: <http://creativecommons.org/ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<https://www.amazon.com/Fire-Blood-HBO-Tie-Targaryen/dp/0593598008>
cc:permits cc:DerivativeWorks, cc:Distribution, cc:Reproduction ;
cc:requires cc:Attribution, cc:Notice, cc:ShareAlike ;
dc:creator <http:dsti#boa> ;
dc:date "2024-03-01" ;
dc:format "text/html" ;
dc:language "en" ;
dc:publisher <http://boa.fr> ;
dc:title " Fire & Blood " .

Q6.4 practice:

1. Get the source of the FoaF schema: http://xmlns.com/foaf/spec/index.rdf
2. Find the property `weblog`
3. What are the types of this property?
4. Does it inherit from other properties?
5. What is its signature?

3- Weblog is of type a rdf:Property, owl:ObjectProperty, owl:InverseFunctionalProperty .
4- it a subproperty of foaf:page
5- The domain is foaf:Agent, and its range is foaf:Document .

Q6.5 practice:

1. Find the FOAF-a-Matic web page
2. Use this tool to generate your FOAF profile in RDF/XML
3. Translate it into Turtle, save and give the result in your answers.
4. Add five specific relationships to your FOAF file using RELATIONSHIPS: http://purl.org/vocab/relationship/

Question 3 and 4:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix ns0: <http://webns.net/mvcb/> .
@prefix rel: < http://purl.org/vocab/relationship/> .
<http://www.w3.org/2004/02/skos/core>
  a foaf:PersonalProfileDocument ;
  foaf:maker skos:me ;
  foaf:primaryTopic skos:me ;
  ns0:generatorAgent <http://www.ldodds.com/foaf/foaf-a-matic> ;
  ns0:errorReportsTo <mailto:leigh@ldodds.com> .

skos:me
```

```
  a foaf:Person ;
  foaf:name "Boa jean-Luc Boa Thiemele" ;
  foaf:title "M" ;
  foaf:givenname "Boa jean-Luc" ;
  foaf:family_name "Boa Thiemele" ;
  foaf:mbox_sha1sum "fa3f3a1b5fbc6b2e54210bb31f37eff632b3ede8" ;
  foaf:homepage <http://jlbt.fr> ;
  foaf:depiction <http://www.w3.org/2004/02/skos/picture_of_me> ;
  foaf:phone <tel:0856435678> ;
  foaf:workplaceHomepage <http://work.fr> ;
  foaf:workInfoHomepage <http://work.fr/jlbtwork> ;
  foaf:schoolHomepage <http://www.w3.org/2004/02/skos/dsti> ;
  foaf:knows [
    a foaf:Person ;
    foaf:name "Arnaud" ;
    foaf:mbox_sha1sum "cc497027bb75bd387561154829de85c3b1a1dc2d"
  ], [
    a foaf:Person ;
    foaf:name "sora" ;
    foaf:mbox_sha1sum "55d07b68a1fd57f94099b29d0172e6886fa40e75"
  ], [
    a foaf:Person ;
    foaf:name "Etienne" ;
    foaf:mbox_sha1sum "948e2e1b11036e62c0e75c0fa51d4187daf04260"
  ] ;
  rel:relationshipacquaintanceOf <http://anna/ressources> ;
  rel:relationshipchildOf <http://ressource/Jason> ;
  rel:relationshipemployedBy <http://resource/Alexander> ;
  rel:relationshipknowsByReputation <http://resource/Elon_Musk> ;
  rel:relationshiplivesWith <http://resource/Anna> .
```

Q6.6 What does this mean?

```
:BioRDF2DBLP a void:Linkset;
               void:target :BioRDF;
               void:target :DBLP;
               void:linkPredicate skos:exactMatch;
               void:triples 8936 .
```

BioRDF2DBLP is a linkset formed with bioRDF and DBLP, the link predicate uses the skos:exactMatch and there a 8936 triples that uses the skos:exactMatch to link the data between the two dataset

Q6.7 practice:

1. Connect to the Void Store SPARQL endpoint: http://lod.openlinksw.com/sparql/
2. Find the void:Dataset with a property dct:subject and retrieve their URI and subjects.
3. Find the void:Dataset with a dct:subject "covid19".

2. SELECT * WHERE {?x void:Dataset; ?p ?v .} Limit 1000

SELECT * WHERE f{?x ‹http://pur1.org/dc/terms/subject> ?V .} Limit 1000

Q6.8 What does this mean?

```
ex:plot prov:used ex:stats1998 .
ex:bar-chart prov:wasGeneratedBy ex:plot .
ex:stats1998 a dcat:Distribution ;
             dcat:format [ rdfs:label "CSV" ] ;
             dcat:mediaType "text/csv" .
```

The bar-chart was generated using ex:plot.
Ex:stats1998 has a distribution which is in CSV and the exact mediatype is “text/csv”

Q6.9 What does this mean?

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix dct:  <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://inria.fr/data#> .

:db-employ
  a dcat:Distribution ;
  dcat:downloadURL <http://wimmics.inria.fr/docs/employ-2014.sql> ;
  dct:title "SQL Dump of the employees" ;
  dct:spatial <http://www.geonames.org/6640252> ;
  dct:issued "2015-01-12"^^xsd:date ;
  dct:temporal <http://reference.data.gov.uk/id/year/2014> ;
  dct:publisher <http://inria.fr> ;
  dcat:mediaType "application/sql" ;
  dcat:format [ rdfs:label "SQL" ] ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/fr> ;
  dcat:byteSize "38729"^^xsd:decimal .

:R2RTransform12 prov:used :db-employ ;
                prov:used :R2R-employ-mapping ;
                prov:used <http://xmlns.com/foaf/0.1/> .

:FoaFDump a void:Dataset;
          void:feature <http://www.w3.org/ns/formats/RDF_XML>;
          void:dataDump <http://wimmics.inria.fr/docs/employ-2014.rdf>;
          void:exampleResource <http://ns.inria.fr/fabien.gandon#me> ;
          void:vocabulary <http://xmlns.com/foaf/0.1/>;
          void:triples 12875;
          dct:title "RDF Dump of the employees" ;
          prov:wasGeneratedBy :R2RTransform12 ;
          prov:generatedAtTime "2015-01-14T11:38:27"^^xsd:dateTime ;
          prov:wasDerivedFrom :db-employ .
```

:db-employ: This represents a dataset in the form of a SQL dump of employee data. It contains information such as the download URL, title, spatial and temporal coverage, publisher, media type, format, language, and byte size.

:R2RTransform12: used db-employ, R2R-employ-mapping and <http://xmlns.com/foaf/0.1/> .

:FoaFDump: This represents another dataset, specifically an RDF dump of the same employee data. It was generated using the :R2RTransform12 and is derived from the db-employ.

Q6.10 practice:

1. Here is the namespace of Ontology for Media Resources:
   http://www.w3.org/ns/ma-ont#
   Negotiate the turtle version of the ontology with curl/wget
2. Study the ontology to know if I can use the property hasLanguage on a Person

```
ma:hasLanguage
  a owl:ObjectProperty ;
  rdfs:comment "Corresponds to 'language' in the Ontology for Media Resources. The
language used in the resource. A controlled vocabulary such as defined in BCP 47
SHOULD be used. This property can also be used to identify the presence of sign
language (RFC 5646). By inheritance, the hasLanguage property applies indifferently at
the media resource / fragment / track levels.  Best practice recommends to use to best
possible level of granularity fo describe the usage of language within a media
resource including at fragment and track levels."^^xsd:string ;
  rdfs:domain ma:MediaResource .
```

```
ma:MediaResource
  a owl:Class ;
  owl:disjointWith ma:Rating, ma:TargetAudience ;
  rdfs:comment "An image or an audiovisual media resource, which can be composed of
one or more fragment / track."^^xsd:string .
```

From that information, we deduce that hasLanguage cannot be applied to a Person. It applies to MediaResource and MediaResource is "An image or an audiovisual media resource, which can be composed of one or more fragment / track."

Q6.11 practice:

3. Connect to the LOV directory: https://lov.linkeddata.es/
4. Search for schemas talking about “music artist”.
5. What is the top ontology you find?
6. What is its version number?
7. Is it reused by other ontologies?
8. How many classes and properties does it have?
9. What expressivity does it use? (RDFS, OWL)

- The first ontology I have found is mo:MusicArtist
- Its version nulber is 2.15
- Yes it is reused by other ontologies, like schema, ov,af …
- It has 60 classes and 166 properties
- It uses owl, rdfs, foaf, dcterms …

Questions from the course on other data formats.

Q7.1 What are the triples produced with this mapping and this table?

```
:My_Table rdf:type rr:TriplesMap ;
  rr:subjectMap [  rr:template
"https://www.ietf.org/rfc/rfc{NUM}.txt"; ];
  rr:predicateObjectMap [
    rr:predicateMap [  rr:predicate dc:title  ];
    rr:objectMap [     rr:column "ttl"     ]
  ].
```

ID	NUM	ttl
87	2616	Hypertext Transfer Protocol -- HTTP/1.1
88	2396	Uniform Resource Identifiers (URI): Generic Syntax

We are creating a mapping, and the subject of the triple we are going to creating is going to have https://www.ietf.org/rfc/rfc{NUM}.txt as a URI and the URI is going to use the value of the columns NUM. our predicate will be dc:title and the object will be the value of the columns ttl

Q7.2 What are the triples encoded in this HTML?

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#cathy"
typeof="Person">
 <p> <span property="name">Catherine Faron</span>
  (mail: <span property="mbox">faron@i3s.unice.fr</span>) is a
friend of
  <span property="knows"
resource="http://ns.inria.fr/fabien.gandon#me">Fabien Gandon</span>
 </p>
</div>
```

#cathy a foaf:Person;
foaf:name “Catherine Faron”;
foaf:mbox “faron@i3s.unice.fr” ;
foaf:knows <http://ns.inria.fr/fabien.gandon#me> .

Q7.3 practice:

1. Look at the Web Page
   https://www.w3.org/TR/xhtml-rdfa-scenarios/scenario-2.html
2. Call the translator on this Web page to get Turtle:
   https://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
3. What does the extracted triple say?
4. Repeat the question with the page:
   https://coffeecode.net/rdfa/codelab/exercises/check_3b.html
5. Test the page:
   http://ns.inria.fr/humans/humans_rdfa.html

2.

```
@prefix dc: <http://purl.org/dc/terms/> .

<https://www.w3.org/TR/xhtml-rdfa-scenarios/scenario-2.html  > dc:creator "Paul"@en .
```

3.

The triple says that https://www.w3.org/TR/xhtml-rdfa-scenarios/scenario-2.html has been created by Paul and Paul is written in English.

4.

```
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix schema: <http://schema.org/> .

<https://coffeecode.net/rdfa/codelab/exercises/check_3b.html> rdfa:usesVocabulary
schema: .
[]
  a schema:TechArticle ;
  schema:name "Structured data with schema.org codelab" ;
  schema:image <https://coffeecode.net/rdfa/codelab/exercises/squares.png> ;
  schema:educationalUse "codelab" ;
  schema:author <http://example.com/AuthorName> ;
  schema:datePublished "20140129" ;
  schema:description """
    About this codelab
""" ;
  schema:articleBody """
    Exercise 1: From basic HTML to RDFa: first steps
    Exercise 2: Embedded types
    Exercise 3: From strings to things
""" .
```

The triples says that https://coffeecode.net/rdfa/codelab/exercises/check_3b.html is using the vocalulary from schema. A blank node is a Tech article named "Structured data with schema.org codelab" has an image at https://coffeecode.net/rdfa/codelab/exercises/squares.png, it is use for codelab , the author of is http://example.com/AuthorName, the published date is "20140129" and it is describes as follow : “”””about this codelab”””””
And has for articleBody """
   Exercise 1: From basic HTML to RDFa: first steps
   Exercise 2: Embedded types
   Exercise 3: From strings to things
""" .

5.

```
@prefix xhv: <http://www.w3.org/1999/xhtml/vocab#> .

<http://ns.inria.fr/humans/data#Eve>
  a xhv:Lecturer, xhv:Person ;
  xhv:name "Eve" ;
  xhv:hasFriend <http://ns.inria.fr/humans/data#Alice> ;
  xhv:hasSpouse <http://ns.inria.fr/humans/data#David> .
```

Q7.4 Use the online tool to play with RDFa adding for instance a “creator” property
https://rdfa.info/play/

Paste your HTML+RDFa code in here, or click on an example above.
A preview of the page will appear to the right with a data visualization below.
You can also see the raw data by clicking on the "Raw Data" tab.

You can learn more about RDFa by reading the

```
  <a property="url" href="http://www.w3.org/TR/rdfa-primer/">
    <span property="name">RDFa 1.1 Primer</span>
    <span property="creator">Antony Usyk</span>
  </a>.
</span>
```

Q7.5 IMDB uses RDFa – OGP for the I like button

1. Choose a movie on IMDB http://www.imdb.com
2. Copy the URL of the page of the movie
3. Go to the RDFa 1.0 RDFa Distiller and Parser: https://www.w3.org/2007/08/pyRdfa/
4. Open the URI option, past the URL of the movie page and configure and perform the extraction to get Turtle
5. Try also the transformation on a translator:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/

4.

```
@prefix og: <http://ogp.me/ns#> .
@prefix ns0: <twitter:> .
@prefix ns1: <imdb:> .

<https://www.imdb.com/title/tt13452446/?ref_=hm_top_tt_i_4>
  og:url "https://www.imdb.com/title/tt13452446/"@en-US ;
  og:site_name "IMDb"@en-US ;
  og:title "Damsel (2024) ⭐ 6.2 | Action, Adventure, Fantasy"@en-US ;
  og:description "1h 50m | 12"@en-US ;
  og:type "video.movie"@en-US ;
  og:image "https://m.media-
amazon.com/images/M/MV5BODRiMTA4NGMtOTQzZC00OWFjLWFmODctMjY2ZTcwYjI5NDMyXkEyXkFqcGdeQX
VyMDc5ODIzMw@@._V1_FMjpg_UX1000_.jpg"@en-US ;
  og:image:height "1250"@en-US ;
  og:image:width "1000"@en-US ;
  og:locale "en_US"@en-US ;
  og:locale:alternate "es_ES"@en-US, "es_MX"@en-US, "fr_FR"@en-US, "fr_CA"@en-US,
"it_IT"@en-US, "pt_BR"@en-US, "hi_IN"@en-US, "de_DE"@en-US ;
  ns0:site "@IMDb"@en-US ;
  ns0:title "Damsel (2024) ⭐ 6.2 | Action, Adventure, Fantasy"@en-US ;
  ns0:description "1h 50m | 12"@en-US ;
  ns0:card "summary_large_image"@en-US ;
  ns0:image "https://m.media-
amazon.com/images/M/MV5BODRiMTA4NGMtOTQzZC00OWFjLWFmODctMjY2ZTcwYjI5NDMyXkEyXkFqcGdeQX
VyMDc5ODIzMw@@._V1_FMjpg_UX1000_.jpg"@en-US ;
  ns0:image:alt "Damsel: Directed by Juan Carlos Fresnadillo. With Millie Bobby Brown,
Ray Winstone, Angela Bassett, Brooke Carter. A dutiful damsel agrees to marry a
handsome prince, only to find the royal family has recruited her as a sacrifice to
repay an ancient debt."@en-US ;
  ns1:pageType "title"@en-US ;
  ns1:subPageType "main"@en-US ;
  ns1:pageConst "tt13452446"@en-US .
```

Q7.6 Test JSON-LD online

1. Transform your FOAF profile in JSON-LD with the translator:
   http://www.easyrdf.org/converter
   http://rdf.greggkellogg.net/distiller
   https://issemantic.net/rdf-converter
   http://rdfvalidator.mybluemix.net/
   http://rdf-translator.appspot.com/

2. Use the following online tool to generate different variations of JSON-LD of your profile (expanded, collapsed, flattened, etc.)
   http://json-ld.org/playground/

1. Example of json-LD format


```
[{"@id":"_:b0","@type":["http://xmlns.com/foaf/0.1/Person"],"http://xmlns.com/foaf/0.1
/name":[{"@value":"Arnaud"}],"http://xmlns.com/foaf/0.1/mbox_sha1sum":[{"@value":"cc49
7027bb75bd387561154829de85c3b1a1dc2d"}]},{"@id":"_:b1","@type":["http://xmlns.com/foaf
/0.1/Person"],
```


2.
For example, compacted format.


```
{
 "@graph": [
   {
    "@id": "_:b0",
    "@type": "http://xmlns.com/foaf/0.1/Person",
    "http://xmlns.com/foaf/0.1/mbox_sha1sum": "cc497027bb75bd387561154829de85c3b1a1dc2d",
    "http://xmlns.com/foaf/0.1/name": "Arnaud"
   },
   {
    "@id": "_:b1",
    "@type": "http://xmlns.com/foaf/0.1/Person",
    "http://xmlns.com/foaf/0.1/mbox_sha1sum": "55d07b68a1fd57f94099b29d0172e6886fa40e75",
    "http://xmlns.com/foaf/0.1/name": "sora" }
```


Q7.7 To provide the metadata of a CSV file I can...

- ☐ include them in a special column of the CSV.
- ☐ put them in a file with the same name plus “-metadata.json”.
- ☐ put them in the first line of my CSV file.
- ☐ put them in a file called “csv-metadata.json” in the same directory.
- ☐ add the URL of the metadata file to the content of my CSV file.

put them in a file with the same name plus “-metadata.json”.
put them in a file called “csv-metadata.json” in the same directory.

Q7.8 TV Catalog : Imagine we submit the following call to an LDP platform

```
GET /catalog/tv/ HTTP/1.1
Host: example.org
Accept: text/turtle; charset=UTF-8
```

and we receive the following answer:

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type",
<http://www.w3.org/ns/ldp#DirectContainer>; rel="type"
Allow: OPTIONS,HEAD,GET,POST,PUT
Accept-Post: text/turtle, application/ld+json
Content-Length: 232
ETag: W/"90231678"
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix cat: <http://example.org/vocab/catalog#> .
<> a ldp:DirectContainer;   ldp:membershipResource <#cat>;
ldp:hasMemberRelation cat:hasProduct;
  dcterms:title "Container of the TV descriptions";
  ldp:contains <tv1>, <tv2> .
<#cat> a cat:Catalog;   dcterms:title "Catalog of TVs";   cat:hasProduct <tv1>,
<tv2> .
```

Which ones of the following statements are true?

- ☐ the container is just a basic container.
- ☐ the container is a direct container.
- ☐ the container is an indirect container.
- ☐ the platform accepts the GET calls.
- ☐ the platform accepts the PATCH calls.
- ☐ the platform accepts RDF/XML format.
- ☐ the platform accepts RDF Turtle.
- ☐ the platform accepts RDF JSON-LD.
- ☐ a link `hasProduct` is automatically created between the resource #cat and the resources of this container

the container is a direct container.
the platform accepts the GET calls.
the platform accepts RDF Turtle.
the platform accepts RDF JSON-LD.
a link hasProduct is automatically created between the resource #cat and the resources of this container

LAB SESSIONS

Remember: just like for programming, it is a good practice to write & validate step by step, incrementally, and to start from copy-pasted examples from the course.

Lab session on RDF.

Software requirements

- A real text editor (e.g. Notepad++, Gedit, Sublime Text, Emacs, etc.) do not use Word!
- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- An RDF online translator:
  - http://www.easyrdf.org/converter
  - http://rdf.greggkellogg.net/distiller
  - https://issemantic.net/rdf-converter
  - http://rdfvalidator.mybluemix.net/
  - http://rdf-translator.appspot.com/
- The SPARQL Corese engine (CORESE-GUI) : https://project.inria.fr/corese/

Understand existing data

1, If you haven't done it yet during the course, get the RDF/XML about http://ns.inria.fr/fabien.gandon#me and translate the RDF/XML into Turtle/N3 syntax using one of the online translators.

Code of validated RDF in N3 syntax:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://ns.inria.fr/fabien.gandon>
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://ns.inria.fr/fabien.gandon#me> ;
  foaf:primaryTopic <http://ns.inria.fr/fabien.gandon#me> .

<http://ns.inria.fr/fabien.gandon#me>
  a foaf:Person ;
  foaf:name "Fabien Gandon" ;
  foaf:title "Dr" ;
  foaf:givenname "Fabien" ;
  foaf:family_name "Gandon" ;
  foaf:nick "Bafien" ;
  foaf:mbox <mailto:fabien.gandon@inria.fr> ;
  foaf:homepage <http://fabien.info> ;
  foaf:depiction <http://www-
sop.inria.fr/members/Fabien.Gandon/common/FabienGandonBackground.jpg> ;
  foaf:phone <tel:0492387788> ;
  foaf:workplaceHomepage <http://www.inria.fr/> ;
  foaf:workInfoHomepage <http://fabien.info> ;
  foaf:schoolHomepage <http://www.insa-rouen.fr> ;
  foaf:knows [
    a foaf:Person ;
    foaf:name "Olivier Corby" ;
    foaf:mbox <mailto:olivier.corby@inria.fr> ;
    rdfs:seeAlso <http://www-sop.inria.fr/members/Olivier.Corby/>
  ], [
    a foaf:Person ;
```

```
    foaf:name "Catherine Faron-Zucker" ;
    foaf:mbox <mailto:faron@polytech.unice.fr> ;
    rdfs:seeAlso <http://www.i3s.unice.fr/~faron/>
  ] .
```

In the RDF Turtle file, can you identify all the links between the two resources http://ns.inria.fr/fabien.gandon and http://ns.inria.fr/fabien.gandon#me ? What do they represent?

```
<http://ns.inria.fr/fabien.gandon>
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://ns.inria.fr/fabien.gandon#me> ;
  foaf:primaryTopic <http://ns.inria.fr/fabien.gandon#me> .
```

http://ns.inria.fr/fabien.gandon has been created by <http://ns.inria.fr/fabien.gandon#me>

http://ns.inria.fr/fabien.gandon has for primary topic <http://ns.inria.fr/fabien.gandon#me>

2, Get the Turtle data of Paris on DBpedia.org then in the file find the triple that declares it as a capital in Europe.

```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix yago:   <http://dbpedia.org/class/yago/> .
dbr:Paris       rdf:type        yago:WikicatCapitals ,
                yago:WikicatCapitalsInEurope ,
                yago:WikicatWorldHeritageSitesInFrance ,
                yago:Center108523483 ,
                yago:Capital108518505 ,
                dbo:City ,
                yago:Seat108647945 ,
                yago:Area108497294 ,
                yago:Prefecture108626947 ,
                yago:WikicatPrefecturesInFrance ,
                yago:WikicatCitiesInFrance ,
                owl:Thing ,
                yago:City108524735 .
```

The triple is:

```
dbr:Paris  rdf:type  yago:WikicatCapitalsInEurope .
```

Humans Knowledge Graph and its namespace http://ns.inria.fr/humans/data#

The major part of this practical session is using a small dataset about a few persons.

The namespace of the dataset is http://ns.inria.fr/humans/data#

1. With your Web browser, visit that namespace and spot the age of Gaston in the graph

   Gaston is 102 years old

2. Use the command `curl` or `wget` to obtain the **XML version** of this dataset from the same address and download it in a file named "**humans_data.xml**" then use the W3C RDF online validation service to validate the RDF/XML and see the triples and the graph.

3. Use the command `curl` or `wget` to obtain the **Turtle version** of this dataset from the same address and download it in a file named "humans_data.ttl"

4. What is the namespace used for instances created / resources described in this file?

The namespace used for resources described is: http://ns.inria.fr/humans/data#

5. In the file how is the association between resources described (Gaston, Laura, etc.) and their namespace done i.e. how is the namespace of these resources specified?

It is specified using qualified name: prefix + ":" + local name

6. What is the namespace of the vocabulary used to describe the resources (hasParent, name, etc.) in the dataset and how is it specified?

the namespace of the vocabulary used to describe the resources is http://ns.inria.fr/humans/schema#
it is specified using the empty prefix: @prefix : <http://ns.inria.fr/humans/schema#> .

7. Find *everything* about John in the turtle file, all available information:

John is a person.
John is 37-year-old.
John has Sophie as a parent.
John has a shirt size of 12.
John has a shoe size of 14.
John has a trousers size of 44.
John has Alice as a friend.
John is the child of Harry.
John is the father of Mark.
John has Jennifer as a spouse.

Create RDF

Here is a statement extracted from the course:

"Jen is an engineer woman, 42-year old, married to Seb who is a man with whom she had two children: Anny who is a woman and Steffen who is a man".

1. Use your text editor and write the above statements in RDF in N3 syntax inventing your own vocabulary. Save you file as "Jen.ttl"
2. Use your favorite text or XML editor and write the above statements in RDF in XML syntax reusing the same vocabulary "Jen.rdf"
3. Use the RDF XML online validation service to validate your XML and see the triples https://www.w3.org/RDF/Validator/
4. In the validator use the option to visualize the graph
    - Use the RDF online translator to validate your N3 and translate it into RDF/XML: http://www.easyrdf.org/converter
    - http://rdf.greggkellogg.net/distiller
    - https://issemantic.net/rdf-converter
    - http://rdfvalidator.mybluemix.net/
    - http://rdf-translator.appspot.com/
5. Compare your RDF/XML with the result of the N3 translation.
6. Translate in other formats to see the results.

Code of validated RDF in N3 syntax:

```
@prefix voca: <http://www.life.fr/voc#> .
@prefix : <http://www.dsti.fr/#> . # URI
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .


:Jen rdf:type voca:Engineer, voca:Woman;
    voca:age "42"^^xsd:string;
    voca:haspouse :Seb;
    voca:hasChild "Anny", "Steffen" ;
    voca:name "Jen" .


:Seb rdf:type voca:Man;
    voca:haspouse :Jen;
    voca:hasChild "Anny", "Steffen" ;
    voca:name "Seb" .

:Anny rdf:type voca:Woman;
    voca:name "Anny" .

:Steffen rdf:type voca:Man;
    voca:name "Steffen" .
```

Code of validated RDF in XML syntax:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ns0="http://www.life.fr/voc#">

  <rdf:Description rdf:about="http://www.dsti.fr/#Jen">
    <rdf:type rdf:resource="http://www.life.fr/voc#Engineer"/>
    <rdf:type rdf:resource="http://www.life.fr/voc#Woman"/>
    <ns0:age rdf:datatype="http://www.w3.org/2001/XMLSchema#string">42</ns0:age>
    <ns0:haspouse xml:lang="en">Seb</ns0:haspouse>
    <ns0:hasChild>Anny</ns0:hasChild>
    <ns0:hasChild>Steffen</ns0:hasChild>
    <ns0:name>Jen</ns0:name>
  </rdf:Description>

  <ns0:Man rdf:about="http://www.dsti.fr/#Seb">
    <ns0:haspouse>Jen</ns0:haspouse>
    <ns0:hasChild>Anny</ns0:hasChild>
    <ns0:hasChild>Steffen</ns0:hasChild>
    <ns0:name>Seb</ns0:name>
  </ns0:Man>

  <ns0:Woman rdf:about="http://www.dsti.fr/#Anny">
    <ns0:name>Anny</ns0:name>
  </ns0:Woman>

  <ns0:Man rdf:about="http://www.dsti.fr/#Steffen">
    <ns0:name>Steffen</ns0:name>
  </ns0:Man>

</rdf:RDF>
```

Query your data

Download the Corese.jar library and start it as a standalone application: Run the command " java -jar -Dfile.encoding=UTF8 " followed by the name of the ".jar" archive. Notice that you need java on your machine and proper path configuration.

This interface provides several tabs: (1) the System tab for traces of execution, (2) a SHACL editor tab (3) a Turtle Editor tab and (4) A "+" tab to create as many queries as you want. Load the annotations contained in the file "Jen.rdf" you created and validated before. Click on the "+" tab to create a new query and the interface contains a default SPARQL query:

```
select * where {  ?x ?p ?y}
```

The SPARQL language will be presented in the next course. Just know that this query can find all the triples of your data. Launch the query and check the results.

Lab session on SHACL.

Software requirements

- A real text editor (e.g. Notepad++, Gedit, Sublime Text, Emacs, etc.)
- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The SPARQL Corese engine (Corese-GUI jar file): https://project.inria.fr/corese/
- The human dataset file http://ns.inria.fr/humans/data#
- The SHACL file http://ns.inria.fr/humans/humans_shape.ttl

What is that shape

With you text editor open the file humans_shape.ttl and look at the content

What is the qualified name of the main shape being defined:
:PersonShape
What is the type of that shape:
It is of type NodeShape
What is the target of that shape:
All the node of type Person

Explain in English the constraint it places on the focus node:

```
@prefix :     <http://ns.inria.fr/humans/schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:  <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:PersonShape a sh:NodeShape ;

        sh:targetClass :Person;

        sh:property [
                sh:message "a Person must have a name"@en;
                sh:severity sh:Violation;
                sh:path :name ;
                sh:minCount 1
        ] .
```

The constraint says that all the node person must have at least one property name. If not, the violation of that constraint is considered as a complete violation and the message "a Person must have a name" is display in the validation report.
What is the severity level of that constraint?
The level of that severity is violation

In Corese load the dataset humans_data.ttl (menu “file > load > Dataset”) and this shape (menu “file > load > SHACL”) and run the validation in a query tab (button “SHACL” in a query tab). Explain in English what the report is saying:
There is a violation of the constraint, the person Karl does not have a property name. we see the message “a Person must have a name” display in the report.

Add your constraints

Extend the shape to add a constraint of severity level “Warning” enforcing that a Person should have an age:

In Corese load the dataset humans_data.ttl (menu “file > load > Dataset”) and this shape (menu “file > load > SHACL”) and run the validation in a query tab (button “SHACL” in a query tab). Explain in English what the report is saying:
David, Eve and Laura do not have a property age. So the message that I specified int my code "a Person must have a name” appears in validation report to tell us that

Extend the shape to add a constraint of severity level “Info” enforcing that a person’s name should be in English:
In Corese load the dataset humans_data.ttl (menu “file > load > Dataset”) and this shape (menu “file > load > SHACL”) and run the validation in a query tab (button “SHACL” in a query tab). Explain in English what the report is saying:

```
@prefix:     <http://ns.inria.fr/humans/schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:  <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

:PersonShape a sh:NodeShape ;

        sh:targetClass :Person;

        sh:property [
                sh:message "a Person must have a name"@en;
                sh:severity sh:Violation;
                sh:path :name ;
                sh:minCount 1;

        ] ;

        sh:property [
                sh:message "a Person must have an age"@en;
                sh:severity sh:Warning;
                sh:path :age ;
                sh:minCount 1 ;
        ];

        sh:property [
                sh:message "a Person must have a name written in English"@en;
                sh:severity sh:info;
                sh:path :name ;
                sh:languageIn("en")
        ].
```

For Eve, John, William,David,Laura,Mark we see an info specifying that the name must be written in english

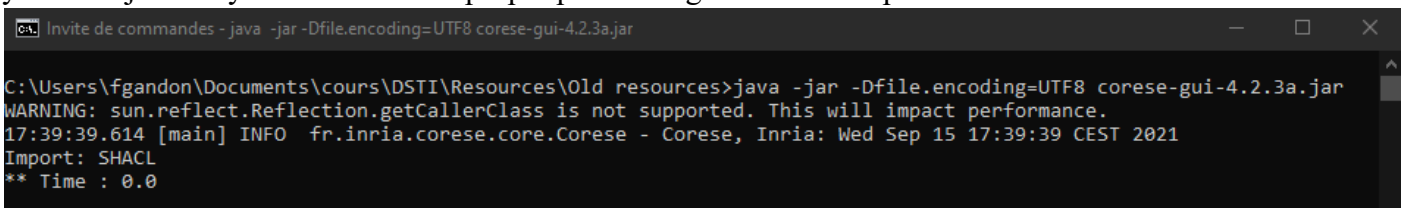
Lab session on SPARQL.

Software requirements

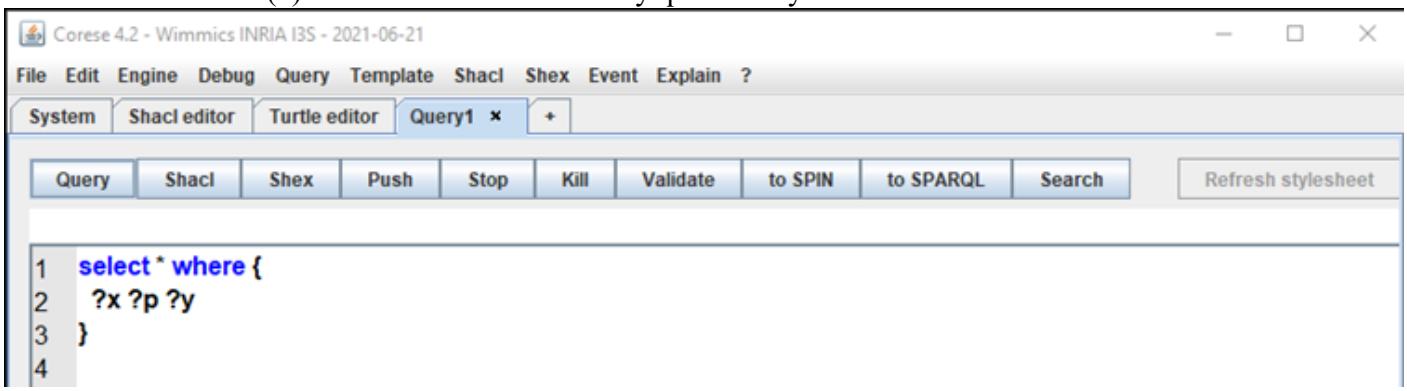
- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The RDF online translator: http://rdf-translator.appspot.com/
- The SPARQL Corese engine (Corese-GUI jar file): https://project.inria.fr/corese/

Basic query on RDF humans dataset

If you haven't done it yet download the SPARQL Corese engine.
Run the command " java -jar -Dfile.encoding=UTF8 " followed by the name of the ".jar" archive. Notice that you need java on your machine and proper path configuration. Example:



This interface provides several tabs: (1) the System tab for traces of execution, (2) a SHACL editor tab (3) a Turtle Editor tab and (4) A "+" tab to create as many queries as you want.



You should have the dataset humans_data.ttl from the previous practical session.
Load the file humans_data.ttl as RDF data in CORESE.

NB: CORESE reads all the formats/syntaxes of RDF.

Question 1:

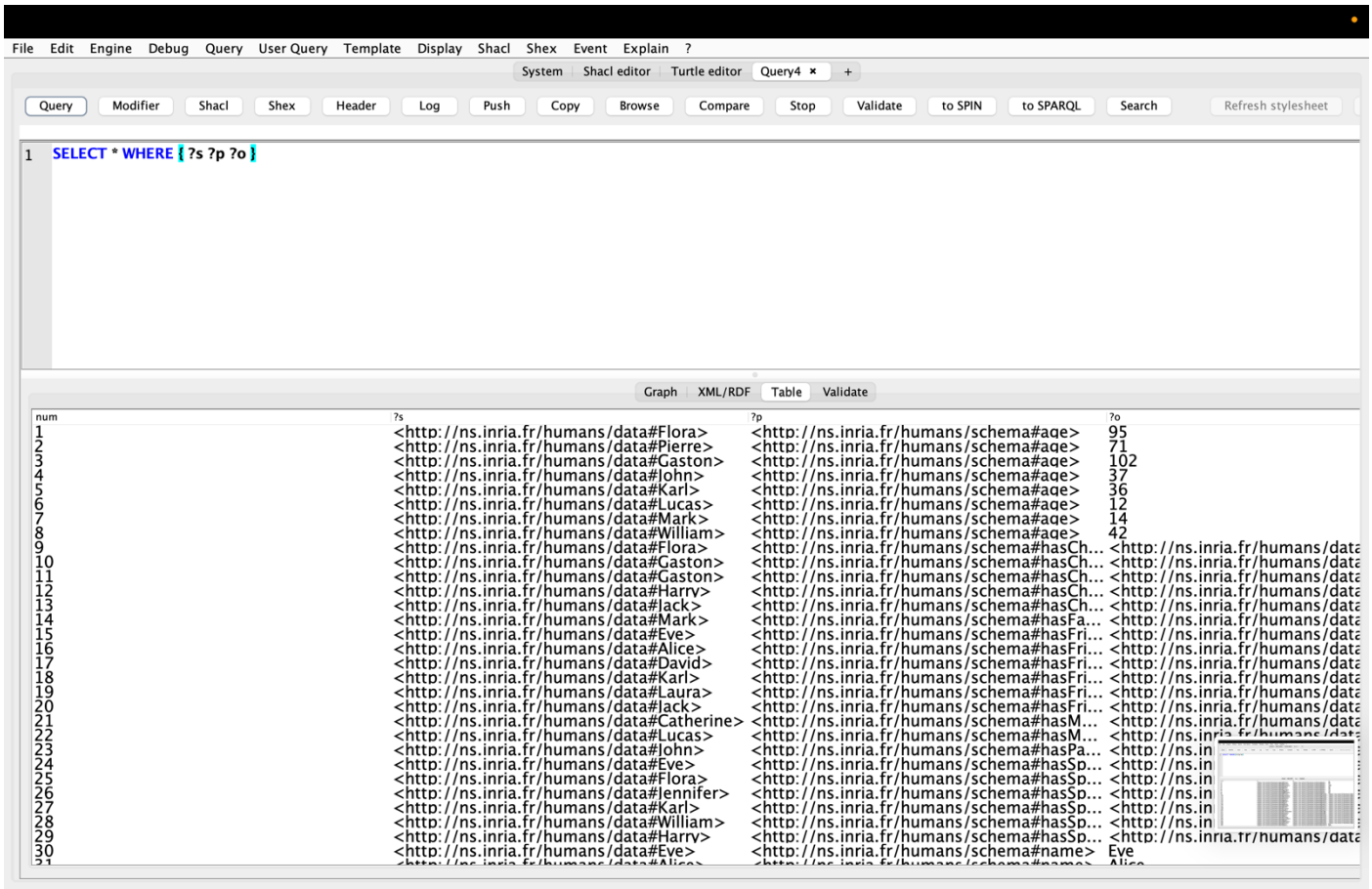
Create a new tab to enter the following query and explain what it does and the results you get. This is a good way to familiarize yourself with the data.

```
SELECT * WHERE { ?s ?p ?o }
```

Explanation:

We are querying for all the resources with all their property and all their value. We are getting as a result the full RDF representation.

Screenshot:



Question 2:

Create a new tab to enter the following query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
select * where { ?x a ?t . filter(strstarts(?t, h:)) }
```

Translate this query in plain English.

We select all triples where the resource ?x has a type (?t). The type ?t must starts the namespace prefix h: .

Run this query. How many answers do you get?

We get 21 answers

Find John and his types in the answers.

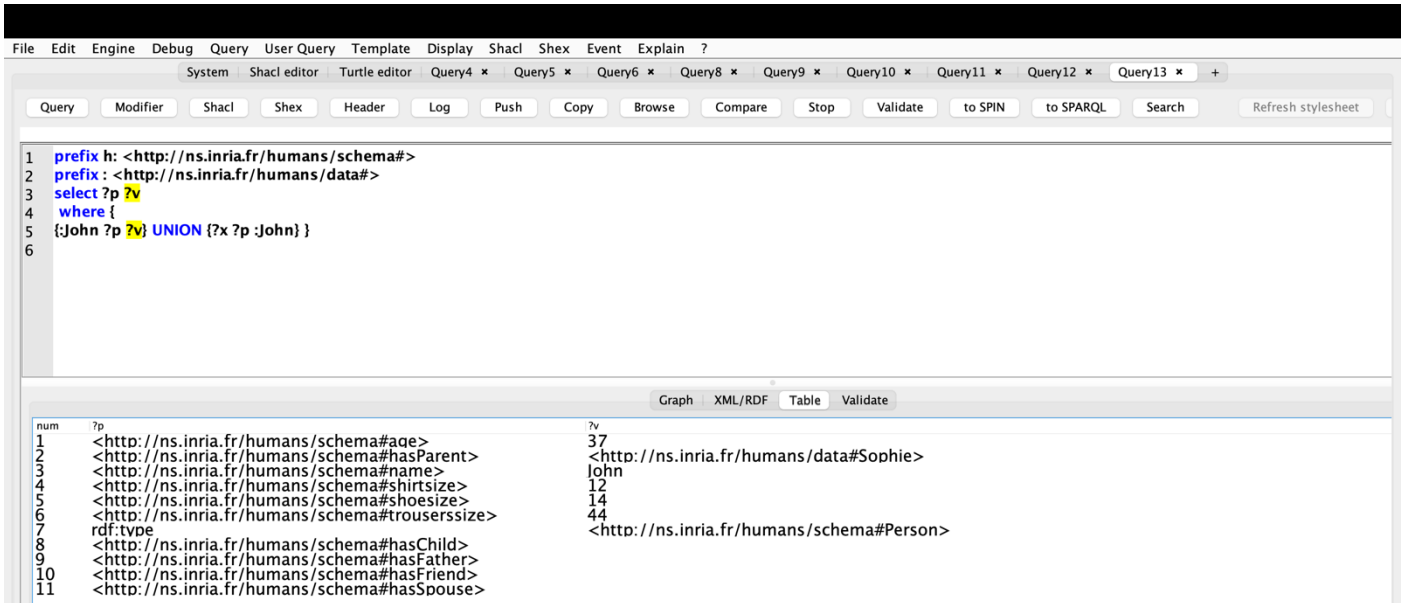
John's types: Person (http://ns.inria.fr/humans/schema#Person)

In the previous answer, locate the URI of John and formulate a SELECT query to find all the properties of John, using his URI

Query

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
select ?p ?y
 where {
{:John ?p ?v} UNION {?x ?p :John} }
```

Results:



Question 3:

Create a new tab to enter the following query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
select * where { ?x h:hasSpouse ?y }
```

Translate this query in plain English.

We select all triples where the resource ?x has the property hasSpouse (which can take any value ?y)

Run this query. How many answers do you get?

We get 6 answers

Question 4:

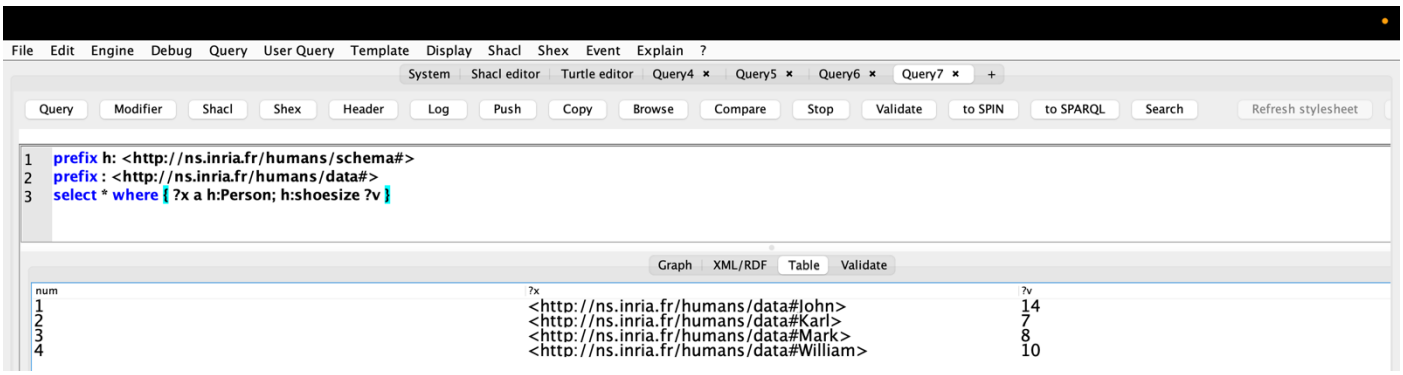
In the RDF file, find the name of the property that is used to give the shoe size of a person.

1. Deduce a query to extract all the persons (h:Person) with their shoe size.

Query:

prefix h: <http://ns.inria.fr/humans/schema#>
prefix: <http://ns.inria.fr/humans/data#>
select * where { ?x a h:Person; h:shoesize ?v }

Result:

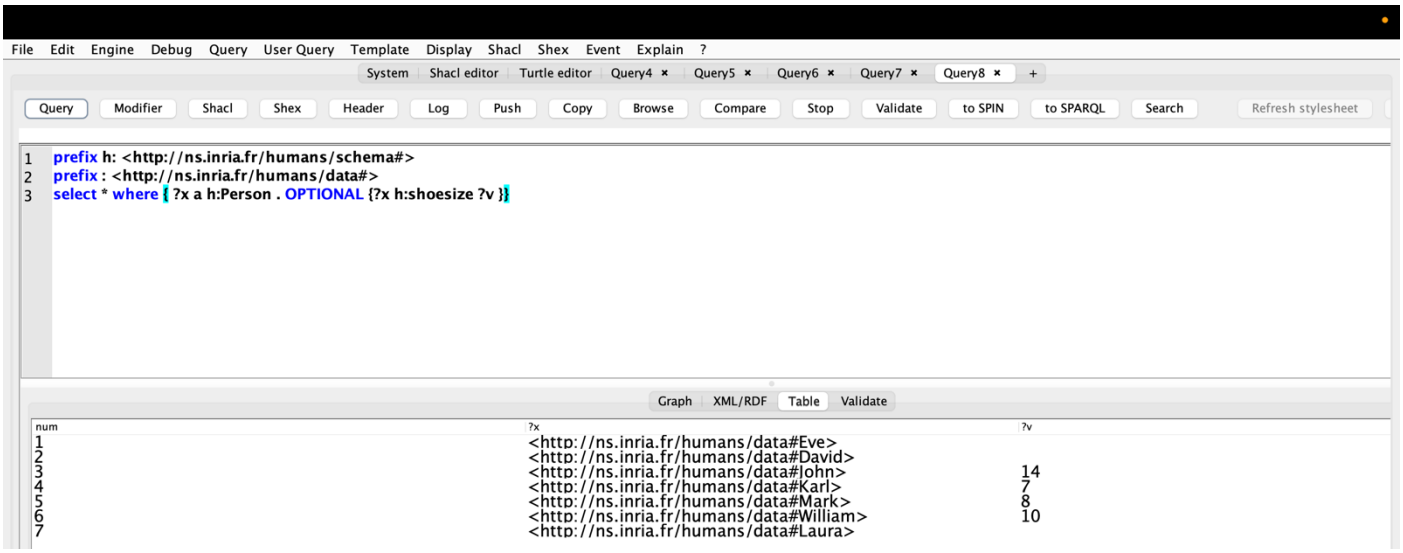


2. Change this query to retrieve all the persons and, if available, their shoe size.

Query:

prefix h: <http://ns.inria.fr/humans/schema#>
prefix: <http://ns.inria.fr/humans/data#>
select * where { ?x a h:Person . OPTIONAL {?x h:shoesize ?v }}

Result:

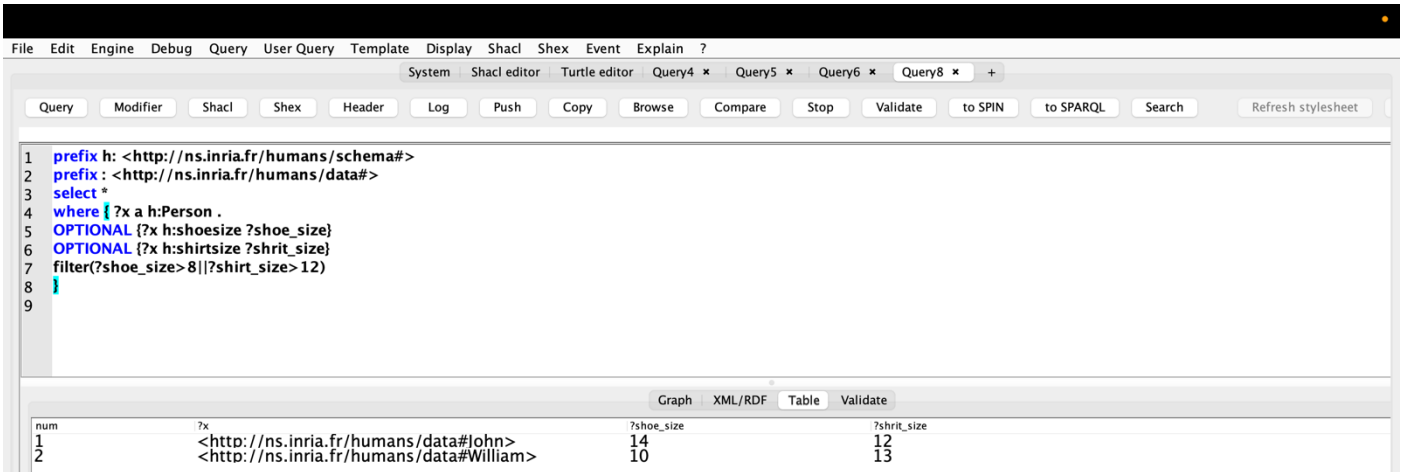


3. Change this query to retrieve all the persons whose shoe size is greater than 8 or whose shirt size is greater than 12.

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
select *
where { ?x a h:Person .
OPTIONAL {?x h:shoesize ?shoe_size}
OPTIONAL {?x h:shirtsize ?shrit_size}
filter(?shoe_size>8||?shirt_size>12)
}
```

Result:



Advanced SPARQL (after second part of the SPARQL course)

Question 5: other clauses than SELECT

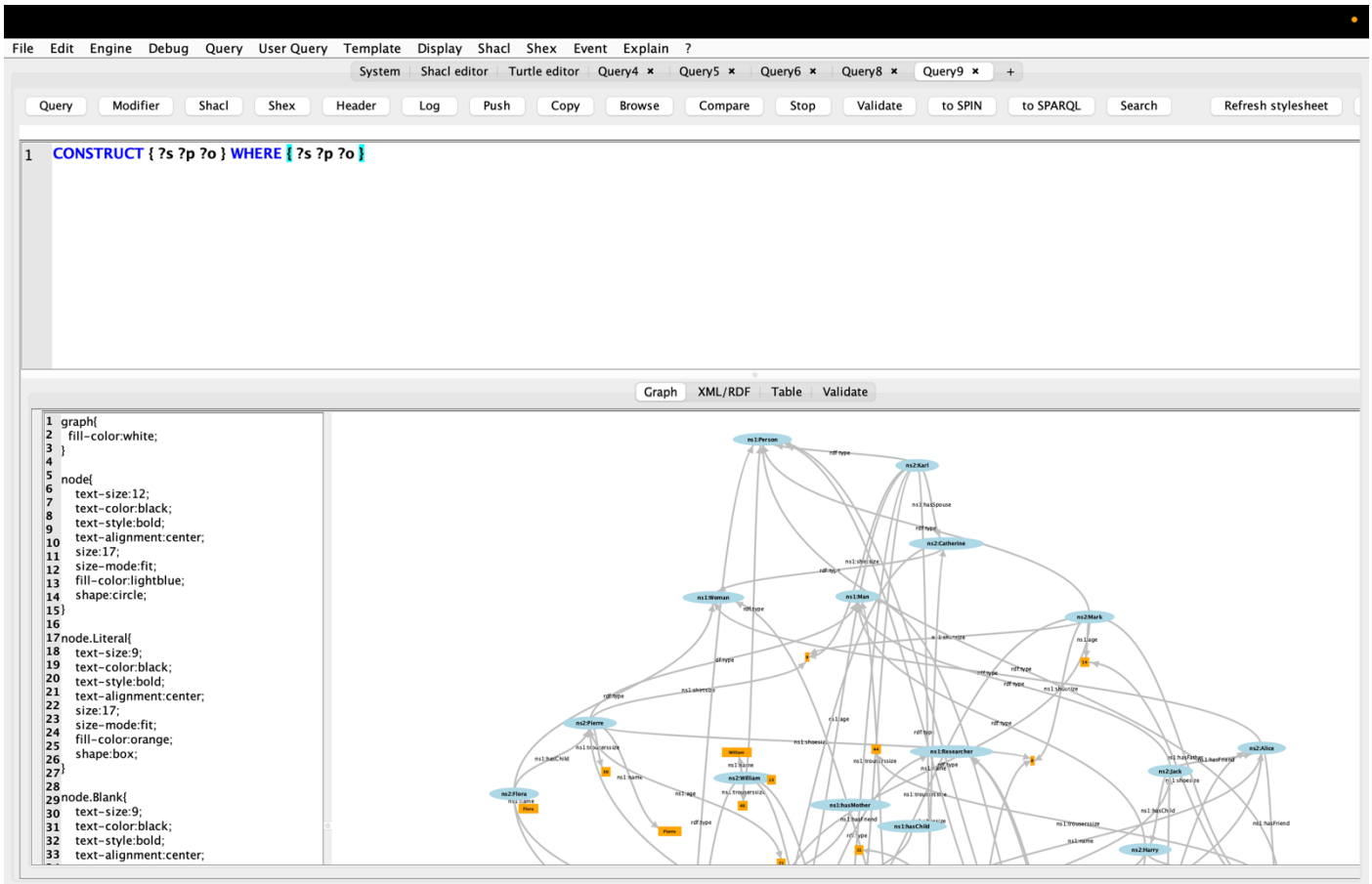
1. Create a new tab to enter the following query and explain what it does and the results you get. This is a good way to familiarize yourself with the data.

```
CONSTRUCT { ?s ?p ?o } WHERE { ?s ?p ?o }
```

Explanation:

We get all the rdf triple and we construct the full graph.

Screenshot:



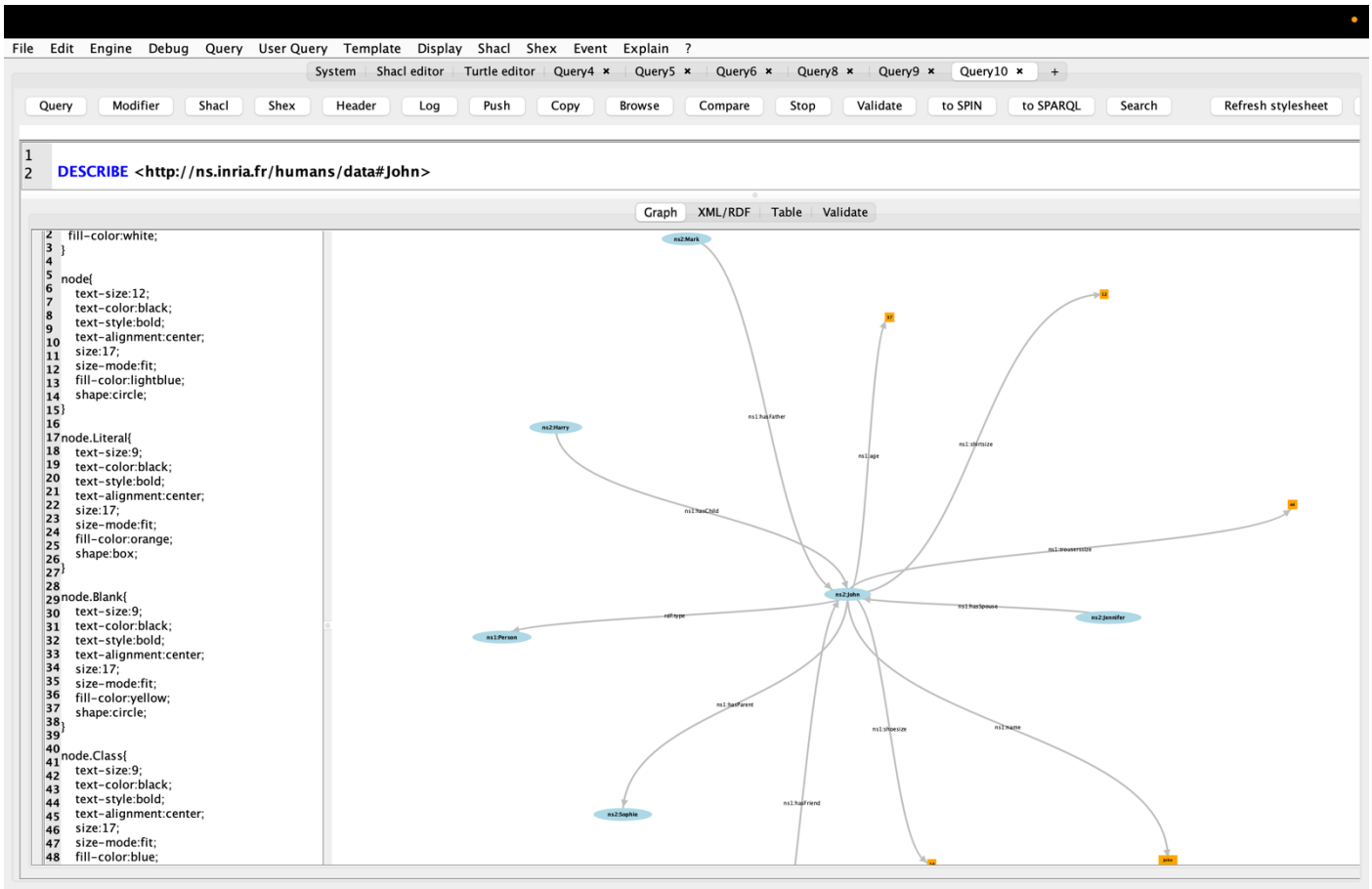
2. In a previous answer you obtained the URI of John. Request a description of John using the special SPARQL clause available for this.

Query

This query asks the SPARQL engine to return RDF triples that describe the resource with the URI.

DESCRIBE <http://ns.inria.fr/humans/data#John>

Results:



Question 6:

In the RDF file, find the name of the property that is used to indicate the children of a person.

1. Formulate a query to find the parents who have at least one child.

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
select ?x ?child_name
where { ?x h:hasChild ?child_name .  }
```

How many answers do you get? How many duplicates do you identify in these responses?

We got five answers and Gaston appears two times.

2. Find a way to avoid duplicates.

Query:

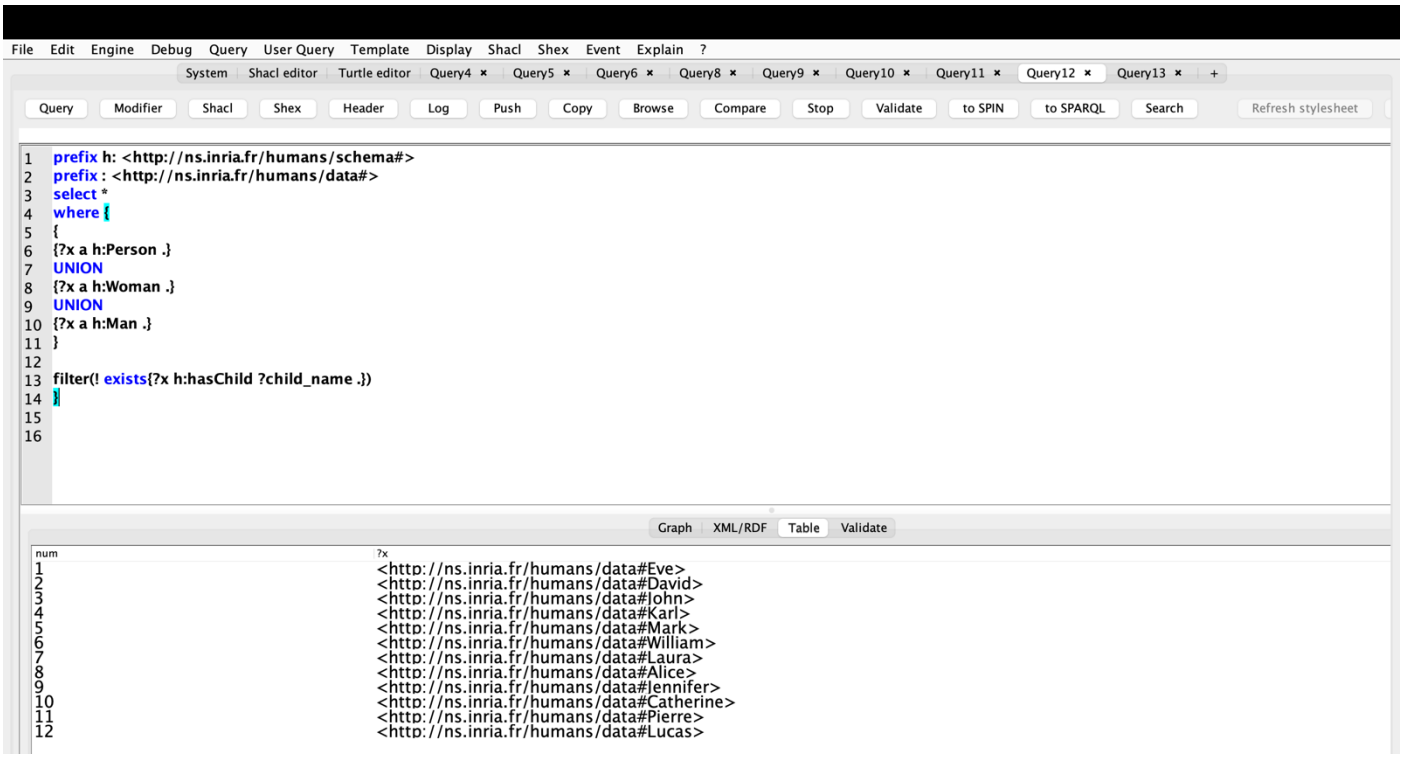
```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
select distinct ?x
where { ?x h:hasChild ?child_name .  }
```

How many answers do you get then?

We got 4 answers.

3. Rewrite a query to find the Persons, Men and Women who have no child.

Query:



Question 7

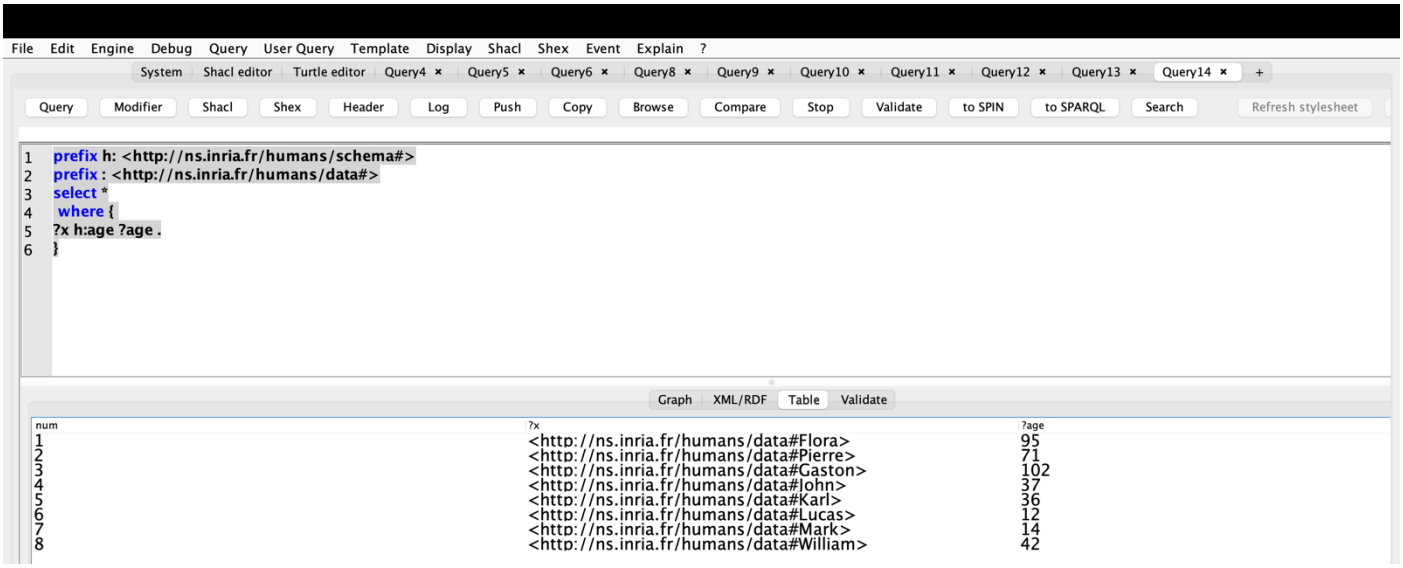
In the RDF file, find the name of the property that is used to give the age of a person.

1. Formulate a query to find persons with their age.

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
select *
 where {
?x h:age ?age .
}
```

Result:



2. Formulate a query to find person who are not adults (here and Adult is a person at least 18 years old).

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
select *
 where {
?x h:age ?age .
```

```
filter(?age<18)
}
```
How many answers do you get?
I got 2 answers

3. Use the appropriate query clause to check if Mark is an adult; use the proper clause statement for this type of query to get a true or false answer.

Query:
```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

select ?name ?age ((?age>=18) as ?is_adult )
where {
?x h:name ?name; h:age ?age .
filter (?x = :Mark)
}
```

4. Write a query that indicates for each person if her age is even (true or false).

Query:
```
prefix h: <http://ns.inria.fr/humans/schema#>
select ?x ?age (if ( abs(floor(?age/2 ) - ceil(?age/2 ) ) =0, True,False) as ?odd)

where { ?x h:age ?age }
```

Question 8

1. **Construct** the symmetric of all hasFriend relations using the good SPARQL statement (ex. When finding `Thomas hasFriend Fabien`, your query should construct `Fabien hasFriend Thomas`)

Query:
```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>
construct {?y h:hasFriend ?x } where {?x h:hasFriend ?y}
```

2. **Insert** the symmetric of all hasFriend relations using the adequate SPARQL statement but check the results with a select query before and after.

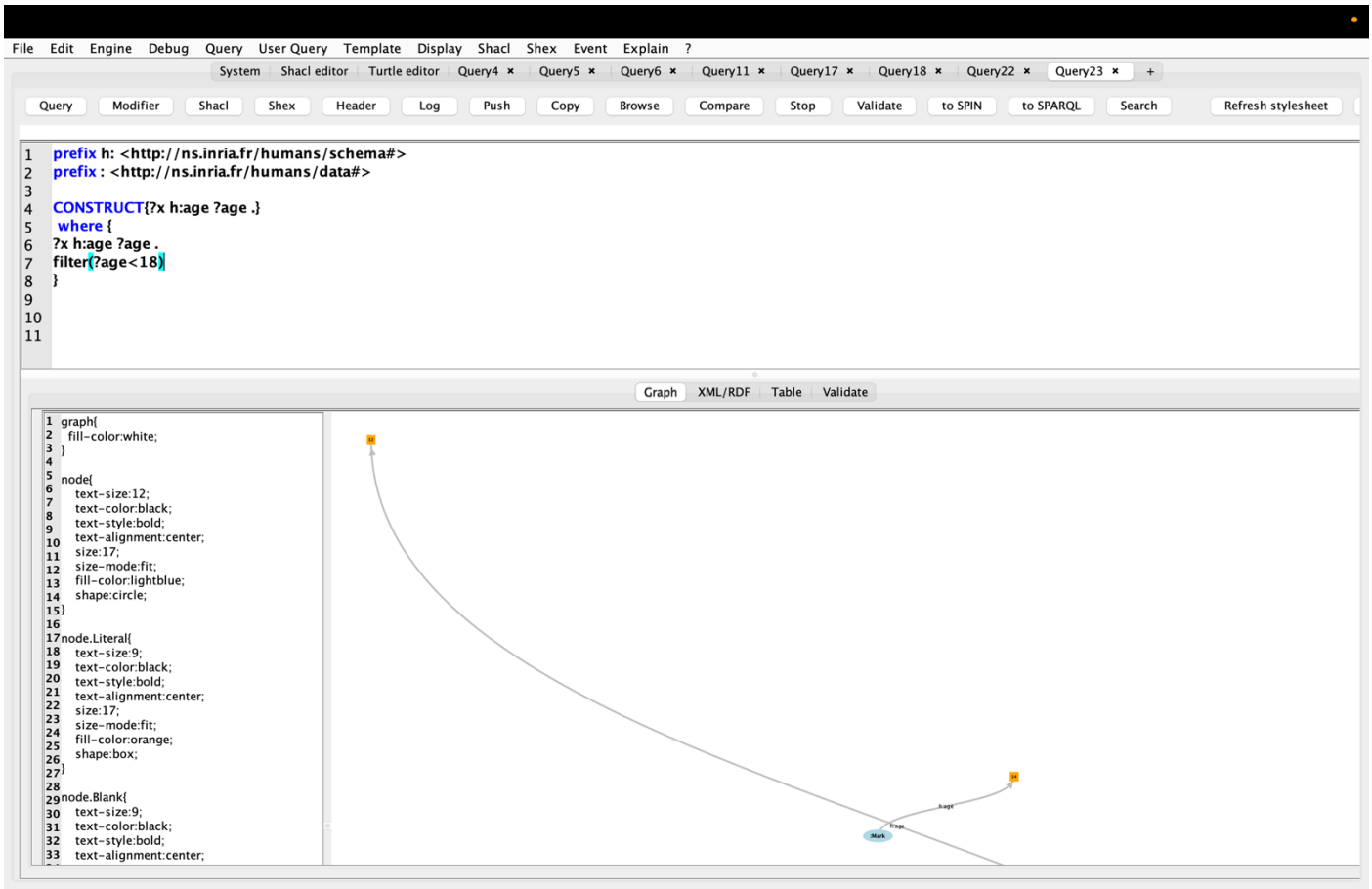
Query:
```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

insert {?y h:hasFriend ?x }
where  { ?x h:hasFriend ?y}
```

Question 9

Choose and edit one of the SELECT WHERE queries previously written to transform them into a CONSTRUCT WHERE query (retaining the same WHERE clause) in order to visualize the results as a graph.

Query:
```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

CONSTRUCT{?x h:age ?age .}
 where {
?x h:age ?age .
filter(?age<18)
}
```

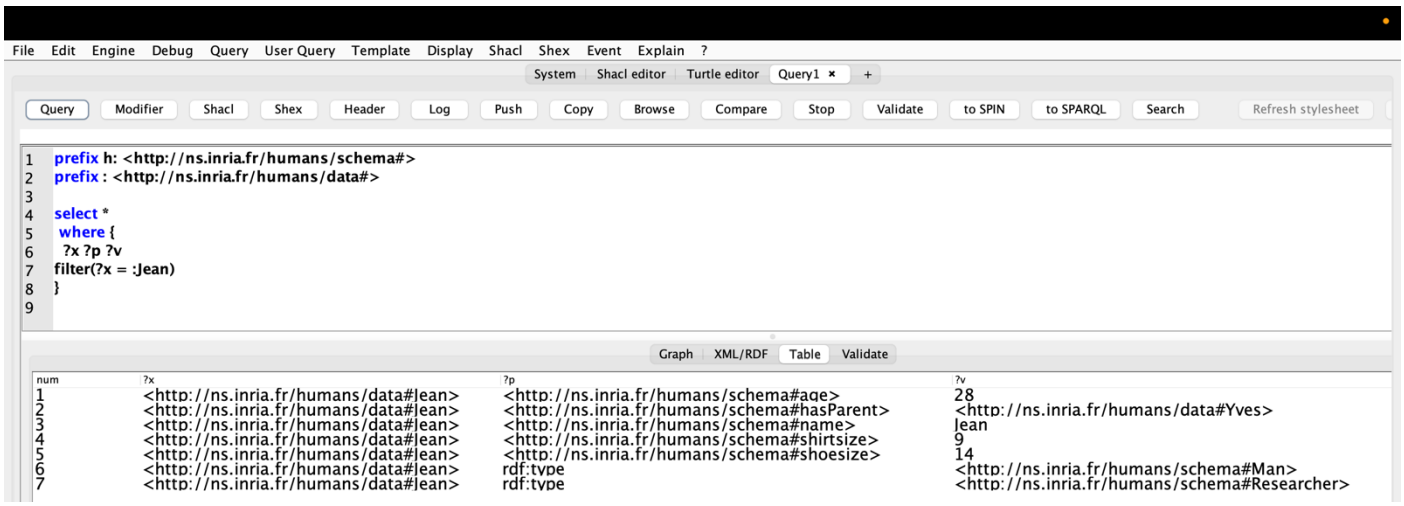
Result:



Question 10

Edit the file to add your own annotation (about you) to the RDF file reusing the properties of the file. Build queries to verify and visualize the annotations you added.

screenshots:



Question 11

1. Formulate a query to find the persons who share the same shirt size.

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

select  ?x ?t ?y
where{?x h:shirtsize ?y .
?t h:shirtsize ?y .
filter(?x<?t)}
```

2. Find the persons who have the same size shirt and construct seeAlso relationship between them.

Query:

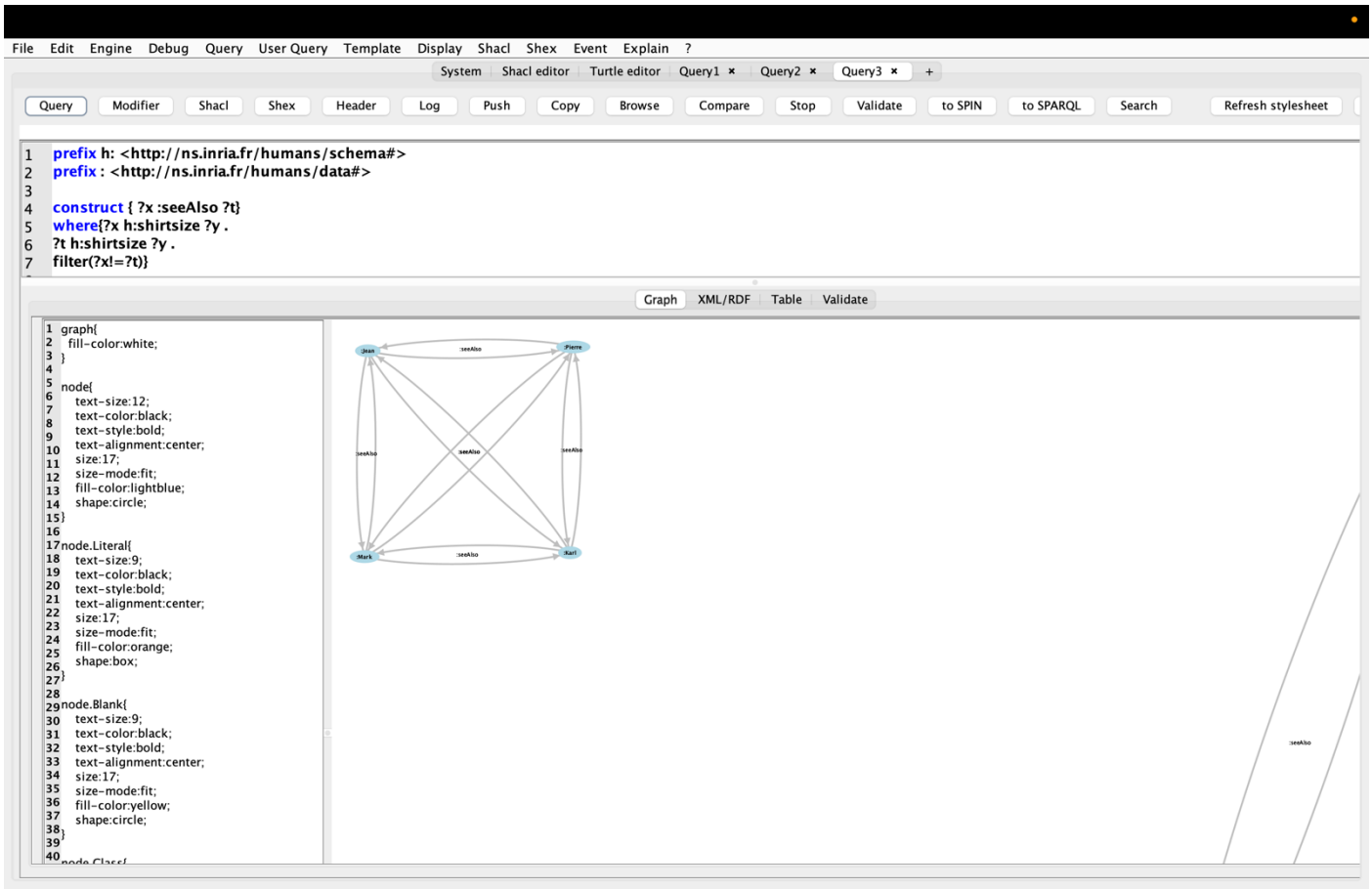
```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

construct { ?x :seeAlso ?t}
where{?x h:shirtsize ?y .
?t h:shirtsize ?y .
filter(?x!=?t)}
```

3. Change the query into an insert.

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

INSERT { ?x :seeAlso ?t}
where{?x h:shirtsize ?y .
?t h:shirtsize ?y .
filter(?x!=?t)}
```

4. Visualize the resources connected by seeAlso (use the CONSTRUCT clause).

screenshot:



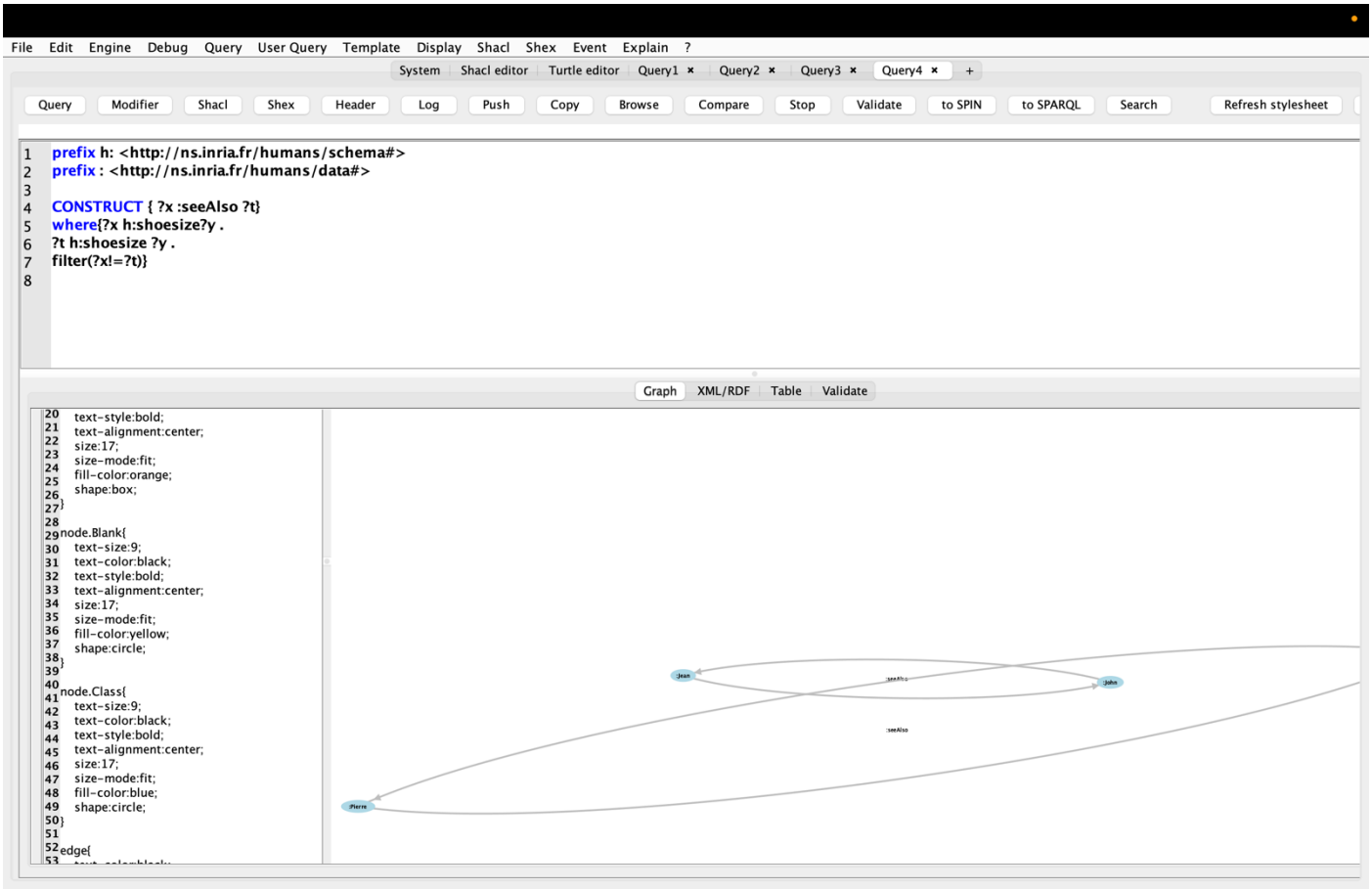
5. Adapt the first query to find persons who have the same shoe size and insert a seeAlso relationship between them.

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

INSERT { ?x :seeAlso ?t}
where{?x h:shoesize?y .
?t h:shoesize ?y .
filter(?x!=?t)}
```

6. Visualize the resources connected by seeAlso (use the CONSTRUCT clause)

screenshot:



7. Change the query to find the resources connected by a path consisting of one or several seeAlso relationships.

Query:

```
select ?x ?y where { ?x h:seeAlso+ ?y }
```

8. Reload the engine (option reload in the menu) and rerun the last visualization query.

Question 12

1. Find the largest shoe size

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

select  (max(?shoe_size) as ?max_shoesize)
where {?x h:shoesize ?shoe_size}
```

2. Find persons who have the biggest size of shoe (subquery + aggregate)

Query:

```
select ?x ?max_shoe
where
{
{
select (max(?shoe) as ?max_shoe)
where { ?person h:shoesize ?shoe }
 }
 ?x h:shoesize ?max_shoe .
```

?x h:name ?name
}

3. Calculate the average shoe size using the appropriate aggregation operator

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

select (avg(?shoe_size) as ?average_shoesize )
where { ?x h:shoesize ?shoe_size}
```

4. Check the average with your own calculation using `sum()` and `count()`

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

select (sum(?shoe_size)/count(?shoe_size) as ?average_shoesize)
WHERE {?x h:shoesize ?shoe_size}
```

Question 13

Find couples without children

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
prefix : <http://ns.inria.fr/humans/data#>

select ?x ?y where
 {?x h:hasSpouse ?y
minus {
{?x h:hasChild ?c1}
union
{?y h:hasChild ?c2}}

}
```

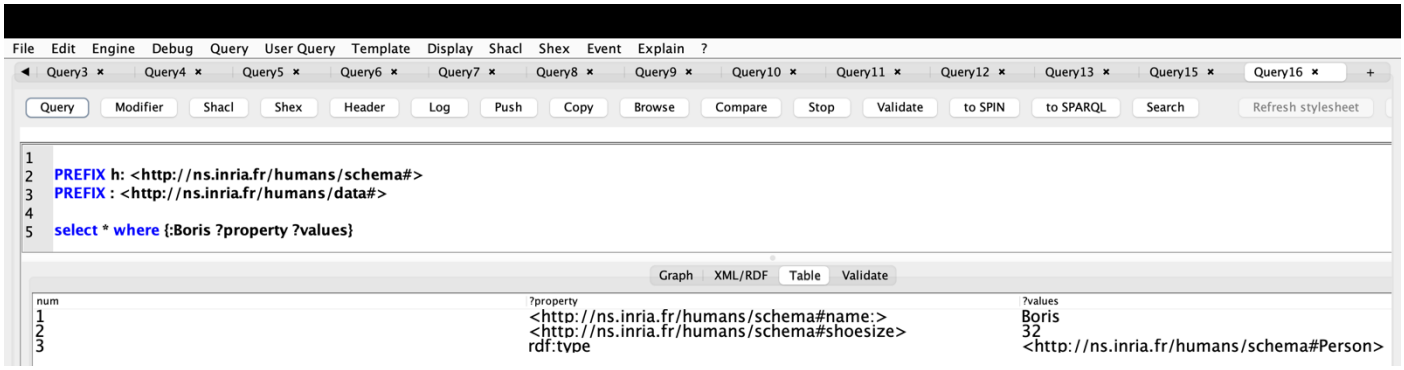
Question 14

Using INSERT DATA, create a new person with its properties. Then, check that it has been created.

Insert:

```
PREFIX h: <http://ns.inria.fr/humans/schema#>
PREFIX : <http://ns.inria.fr/humans/data#>

INSERT DATA {:Boris a h:Person; h:shoesize 32; h:name:"Boris"}
```

Screenshot result:



Question 15

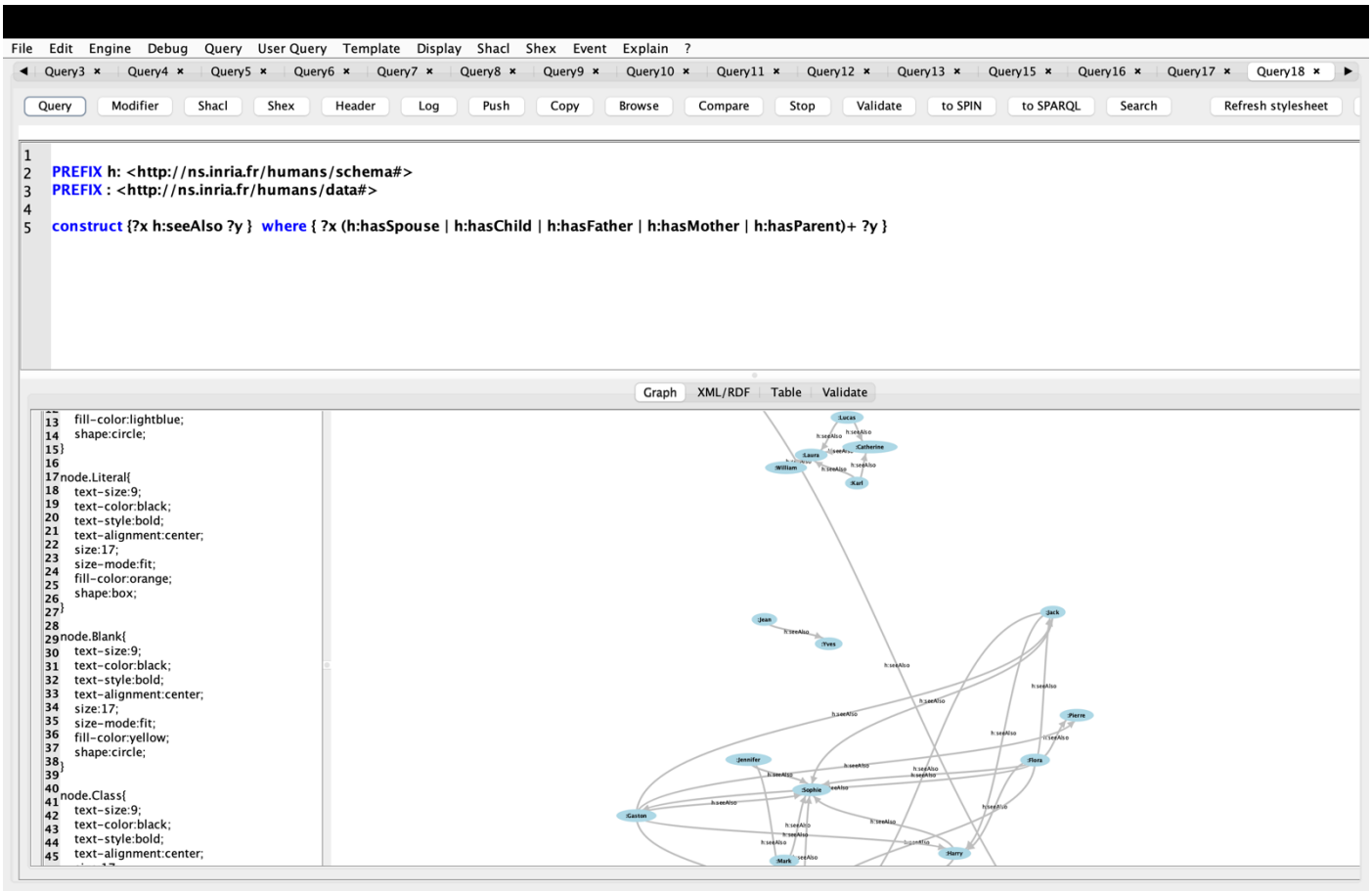
Find the persons connected by paths of any family links. Construct an arc seeAlso between them to visualize the result.

query:

PREFIX h: <http://ns.inria.fr/humans/schema#>
PREFIX : <http://ns.inria.fr/humans/data#>

select ?x ?y where { ?x (h:hasSpouse | h:hasChild | h:hasFather | h:hasMother | h:hasParent)+ ?y }

screenshot:



Question 16

Run the following query:

```
prefix db: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix h: <http://ns.inria.fr/humans/schema#>
construct  { ?x h:name ?nx . ?y h:name ?ny . ?x h:hasSpouse ?y }
where {
  service <http://fr.dbpedia.org/sparql/> {
     select * where {
       ?x db:spouse ?y .
       ?x foaf:name ?nx .
       ?y foaf:name ?ny .
     }
     limit 20
  }
}
```

Explain what it does

We access remotely to a SPARQL endpoint at http://fr.dbpedia.org/sparql/ and we run the query at the endpoint. It retrieves information about the 20 first pairs of spouses and their names from DBpedia. Then, it constructs triples for each pair.

modify it to insert new persons in the base and check the results.

query:

prefix db: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>

```
prefix h: <http://ns.inria.fr/humans/schema#>

insert { ?x h:name ?nx . ?y h:name ?ny . ?x h:hasSpouse ?y } where {
service <http://fr.dbpedia.org/sparql/> { select * where {
        ?x db:spouse ?y .
        ?x foaf:name ?nx .
        ?y foaf:name ?ny .
}
limit 20 }
}
```

Lab session on RDFS.

Software requirements

- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The RDF online translator: http://rdf-translator.appspot.com/
- The SPARQL Corese engine (Corese-GUI jar file): https://project.inria.fr/corese/

About the humans schema

1. If you don't have the human schema file yet use the command curl or wget to obtain the Turtle version of this schema from its namespace and download it in a file named "humans_schema.ttl"

   http://ns.inria.fr/humans/schema#

2. What is the namespace of this ontology? How was it specified?

   The namespace of the rdfs ontology is: http://www.w3.org/2000/01/rdf-schema#
   It is specified using qualified name.

3. Locate the use of the terms of the RDF (S) language: Class, Property, label, comment, range, domain, subClassOf, subPropertyOf, etc. What are their namespaces?

   For the class, the namespace is http://ns.inria.fr/humans/schema#

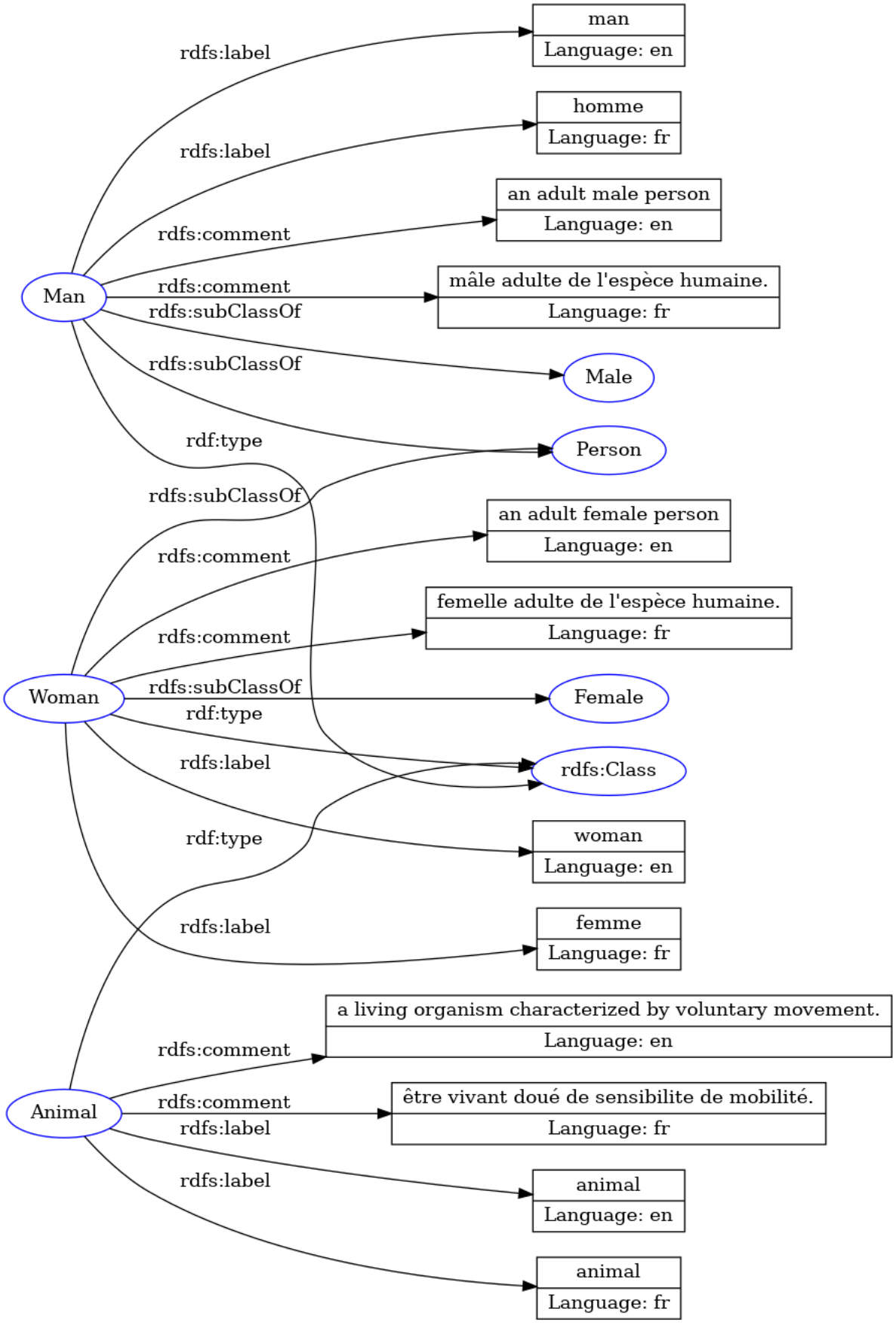
   For the subClassOf, domain, range, label, comment, subPropertyOf: http://www.w3.org/2000/01/rdf-schema#

   For property it is: http://www.w3.org/1999/02/22-rdf-syntax-ns#
   For the

4. What are the classes of resources that can have the age property? Explain
   Any resources can use this property because there is no signature for the property age. (no domain and range)

5. Look at the beginning of the file and draw the subgraph of the hierarchy containing the classes `Animal`, `Man` and `Woman`.

Drawing of hierarchy:



Namespaces:
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
xsd: http://www.w3.org/2001/XMLSchema#
owl: http://www.w3.org/2002/07/owl#
http://ns.inria.fr/humans/schema#

Query the schema itself

Reset or relaunch the standalone Corese search engine interface and load the file `humans_schema.ttl` (and only this one).

1. Write a query to find all the classes of the ontology.

query:

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

select ?x
 where
{ ?x a rdfs:Class}
```

2. Write a query to find all the links subClassOf in the ontology.

query:

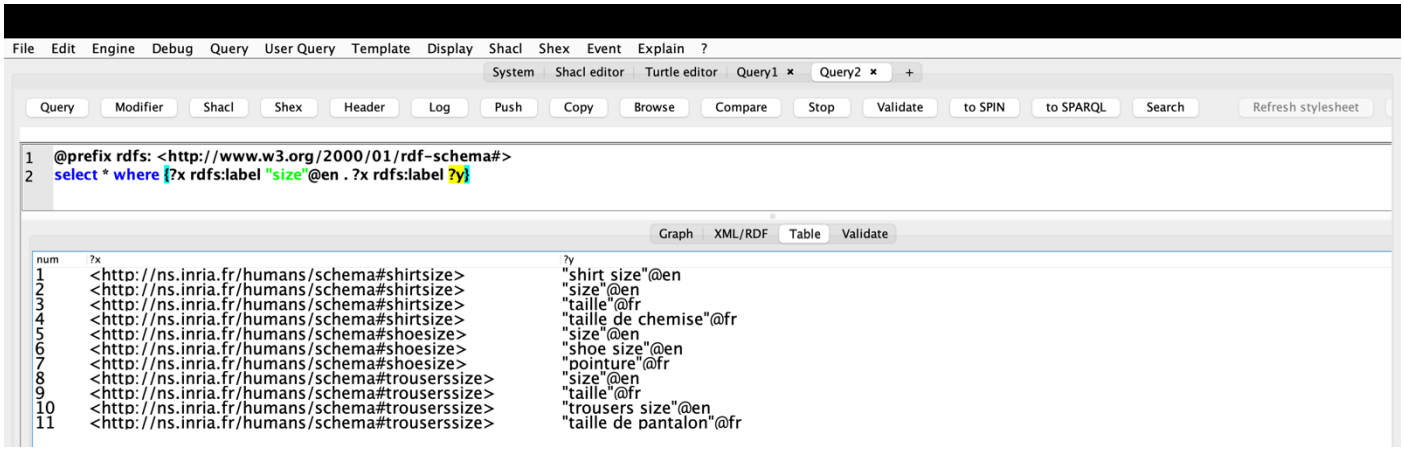
```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

select ?x ?y where { ?x rdfs:subClassOf ?y}
```

3. Write a query to find the definitions and translations of "shoe size" (*other* labels and comments in different languages for the resource labeled "shoe size").

query:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select * where
{?x rdfs:label "size"@en . ?x rdfs:label ?y}
```

answers:



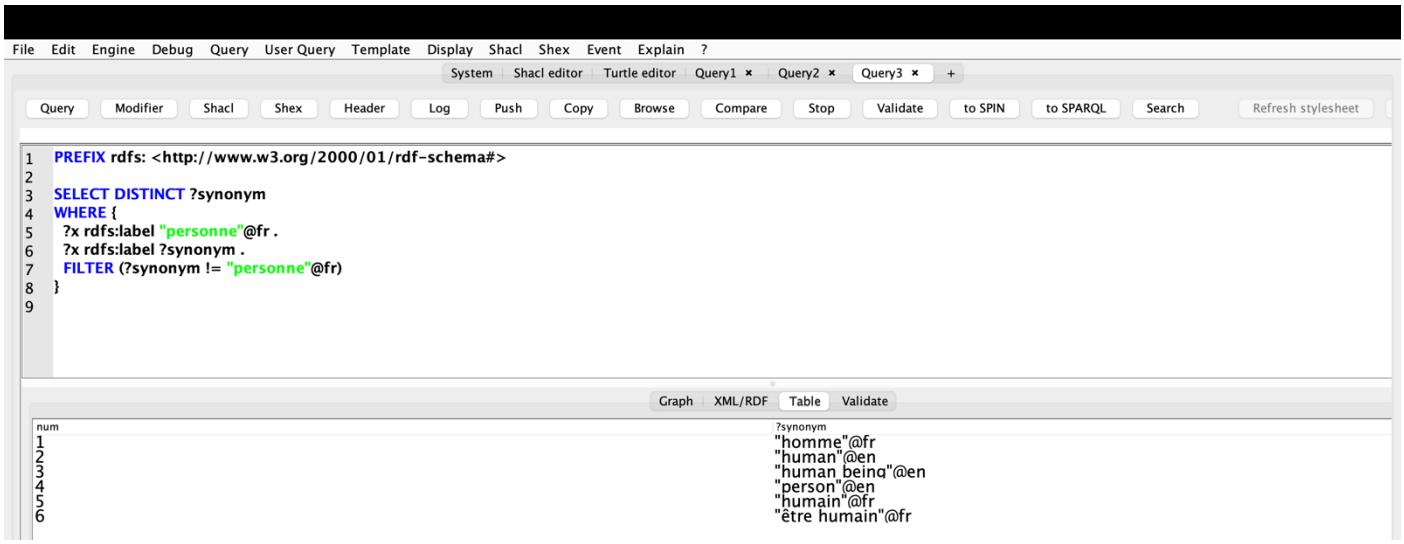
4. Write a query to find the synonyms in French of the word 'personne' in French (*other* labels in the same language for the same resource/class/property). What are the answers?

query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?synonym
WHERE {
  ?x rdfs:label "personne"@fr .
```

```
  ?x rdfs:label ?synonym .
  FILTER (?synonym != "personne"@fr)
}
```

answers:

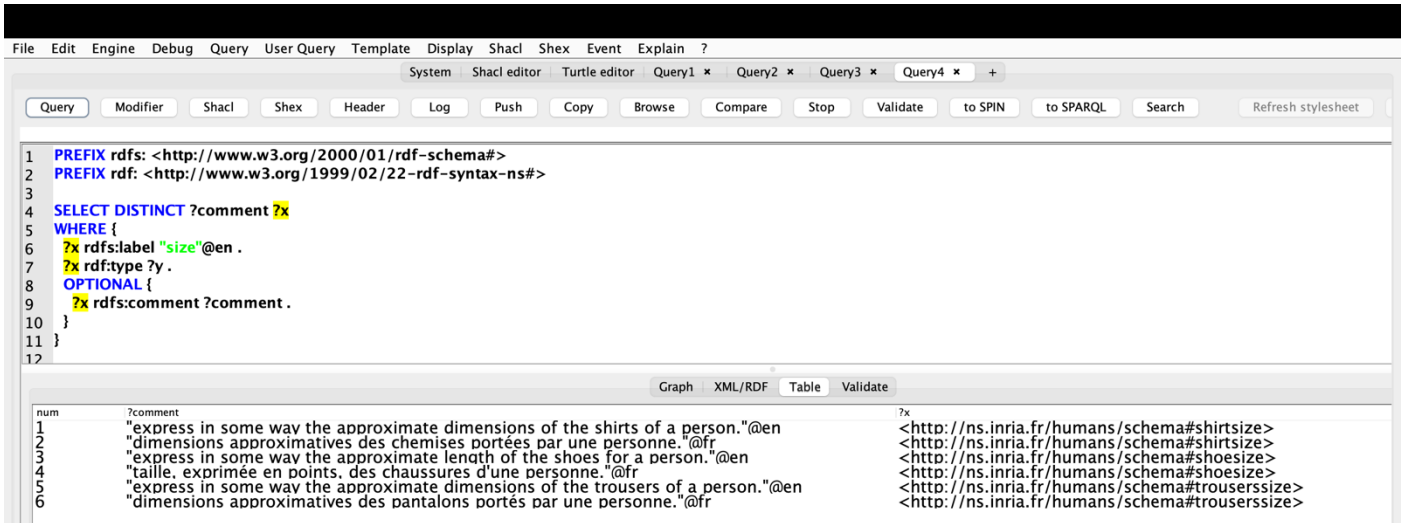


5. Write a query to find the different meaning of the term "size" (disambiguation using the different comments attached to different resources/classes/properties having the label "size"). What are the answers?

query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?comment ?x
WHERE {
  ?x rdfs:label "size"@en .
  ?x rdf:type ?y .
  OPTIONAL {
    ?x rdfs:comment ?comment .
  }
}
```

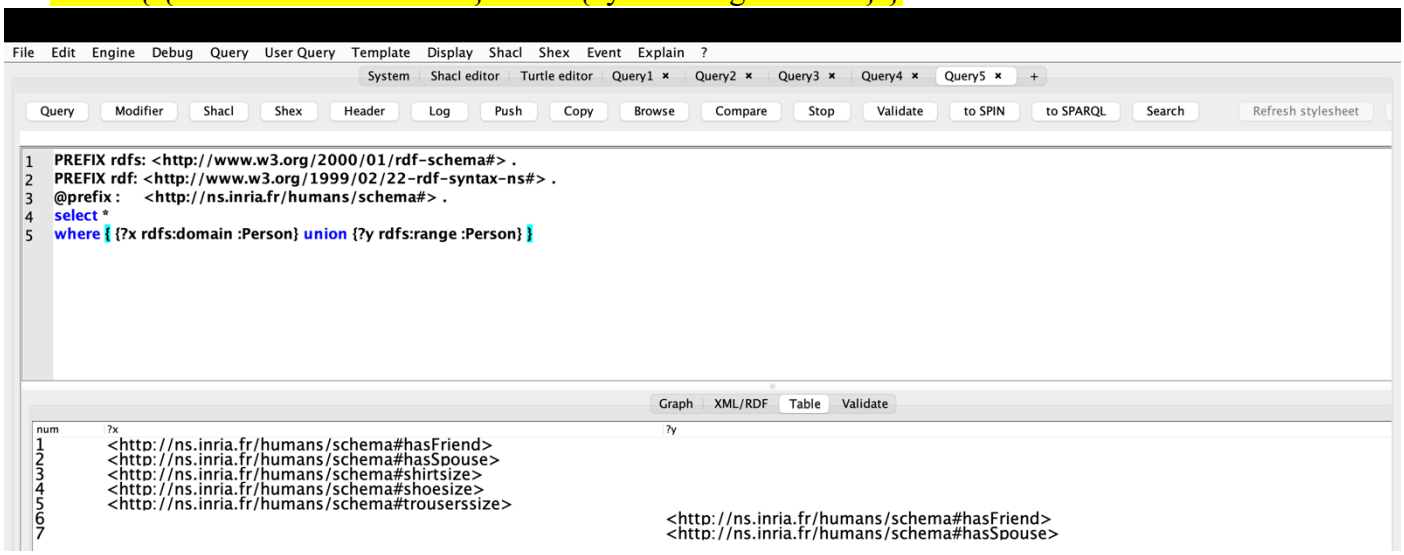
answers:



6. Write a query to find the properties that use the class Person in their signatures?

query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :     <http://ns.inria.fr/humans/schema#> .
select *
where { {?x rdfs:domain :Person} union {?y rdfs:range :Person} }
```

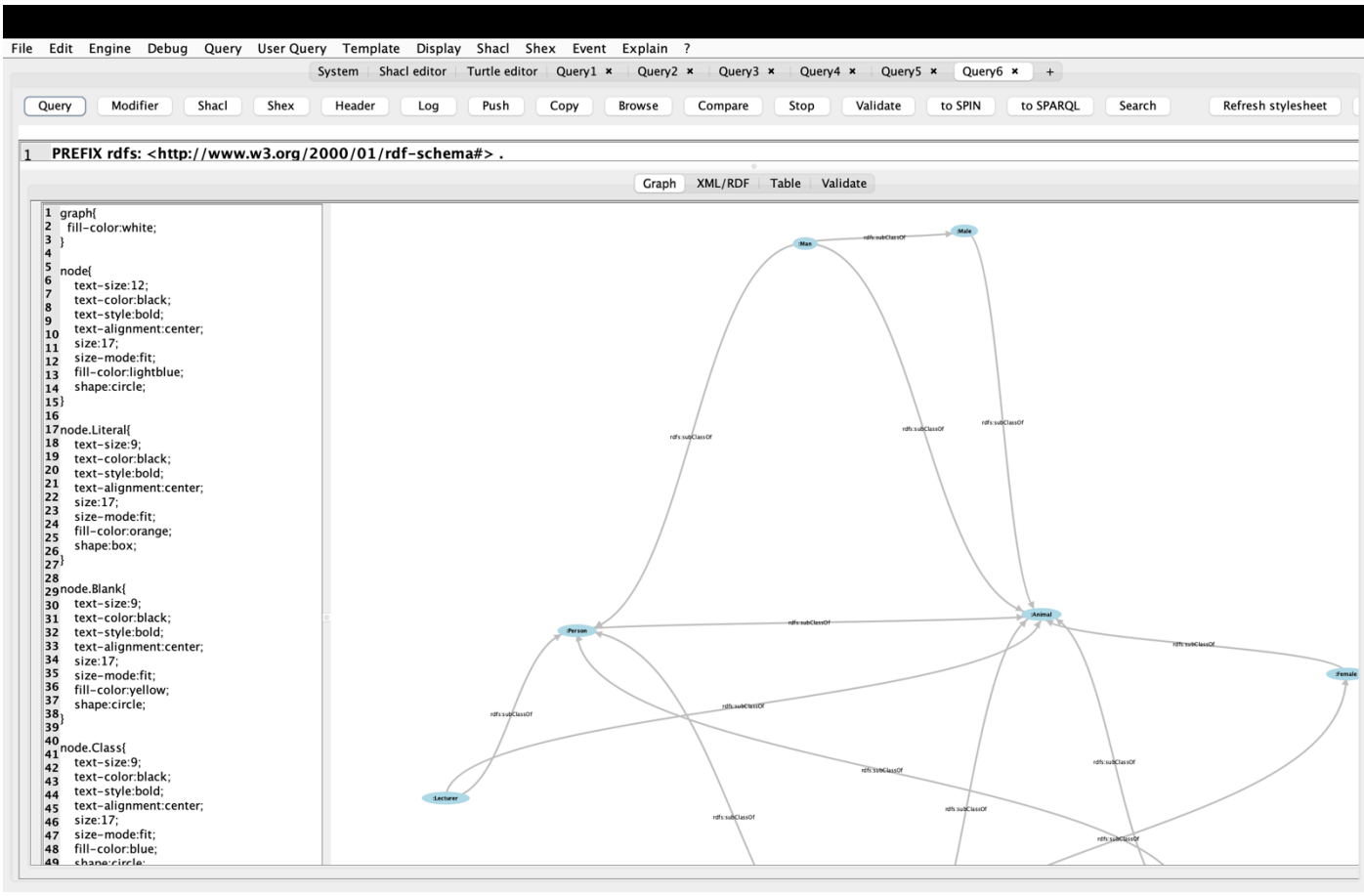


7. Make CORESE draw the graph of the hierarchy of Classes using a CONSTRUCT query considering only the classes in the humans schema

query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :     <http://ns.inria.fr/humans/schema#> .

construct {?x rdfs:subClassOf ?y} where {?x (rdfs:subClassOf)+ ?y }
```

screenshot:

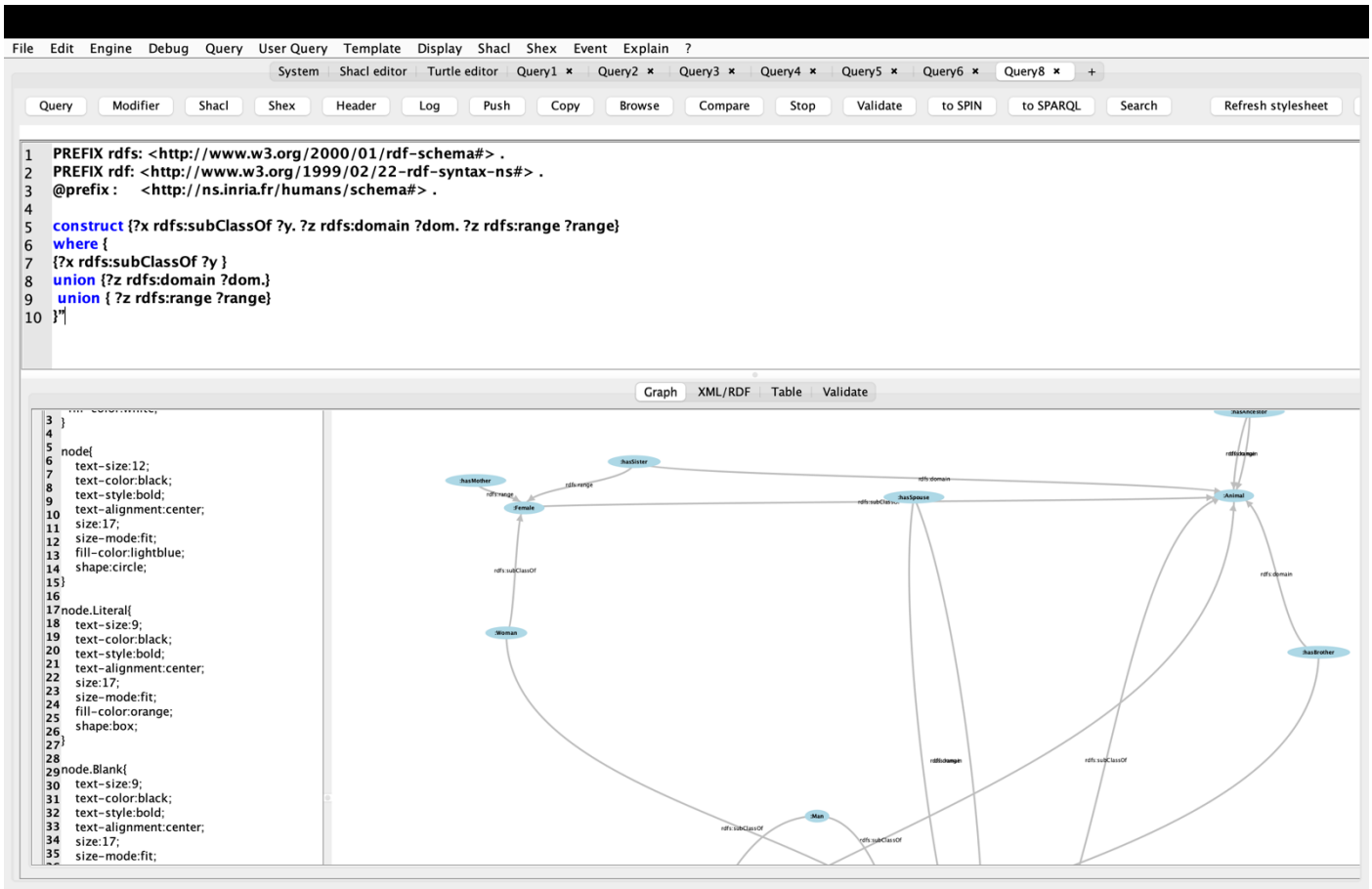
8. 

To the previous CONSTRUCT add the signatures of the relations.

query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://ns.inria.fr/humans/schema#> .

construct {?x rdfs:subClassOf ?y. ?z rdfs:domain ?dom. ?z rdfs:range ?range}
where {
{?x rdfs:subClassOf ?y }
union {?z rdfs:domain ?dom.}
 union { ?z rdfs:range ?range}
}
```

screenshot:



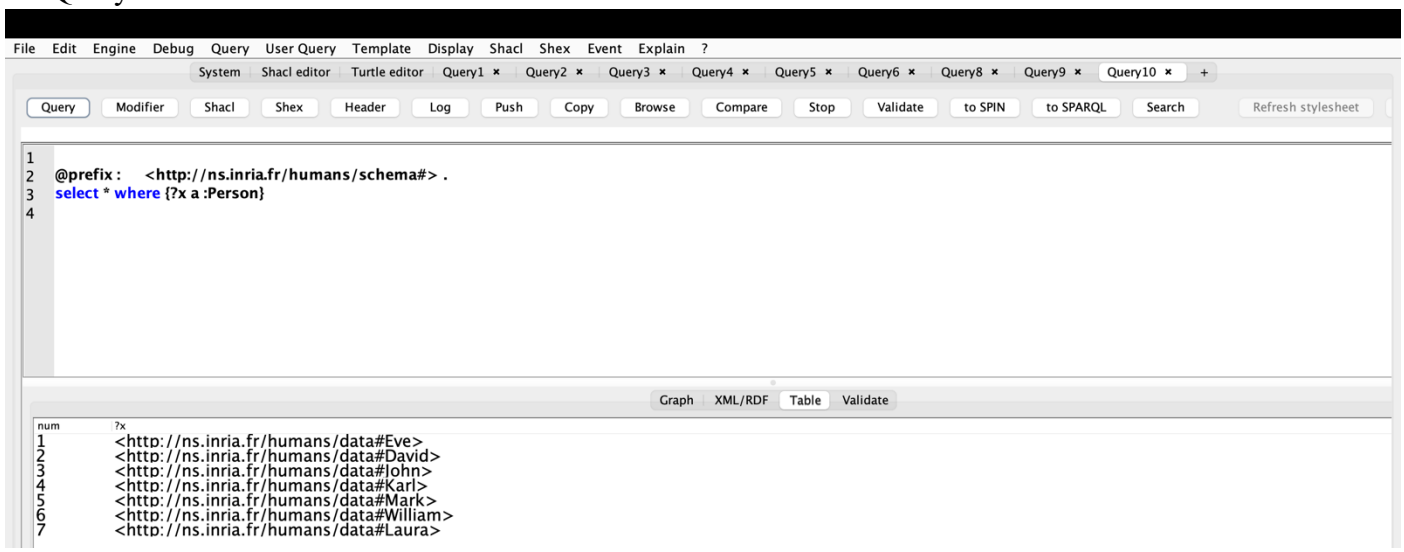
You now know how to query schemas on the semantic Web!

Query data augmented by an RDFS schema

Question 1

1. Reset the Corese engine and load only the data (humans_**data**.ttl)
2. Write a query to find the Persons.

Query:



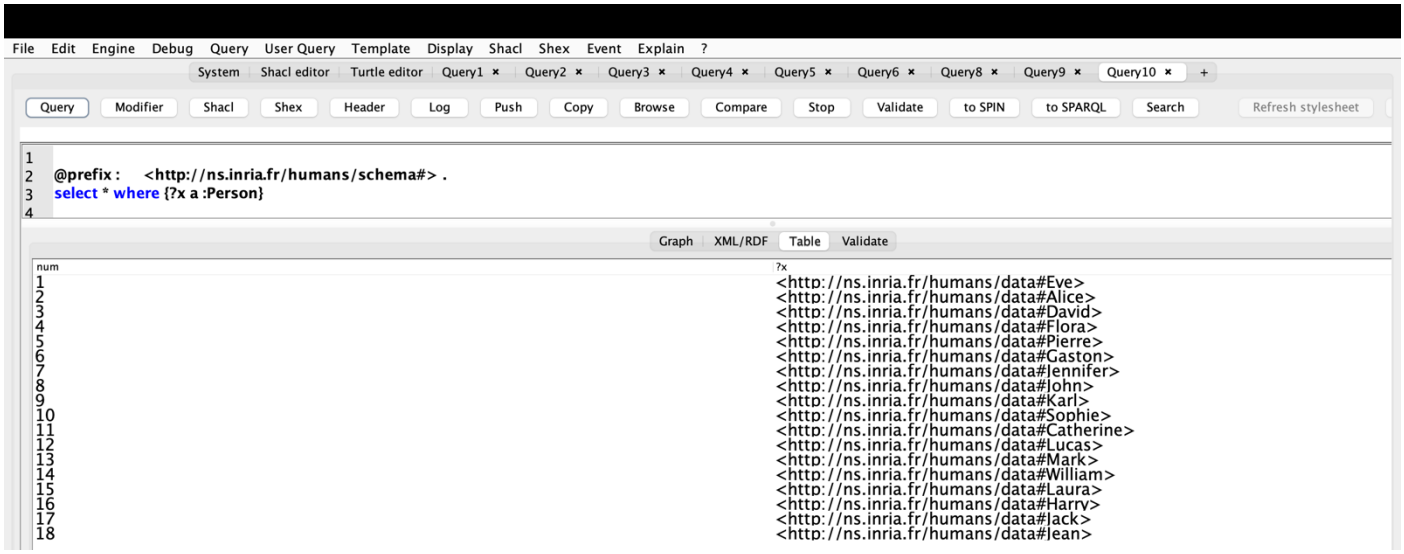
Number of results before:

We got 7 results.

3. Load the schema (humans_**schema**.ttl)
4. Rerun the query to find the Persons and explain the result.

New number of results after and your explanation:

We got 18 results. Loading schemas made inference by the machine possible. It allows to propagate the type :Person to new entities.



Question 2

1. Write a query to find Males and their wives. How many answers do you get? Explain this result.

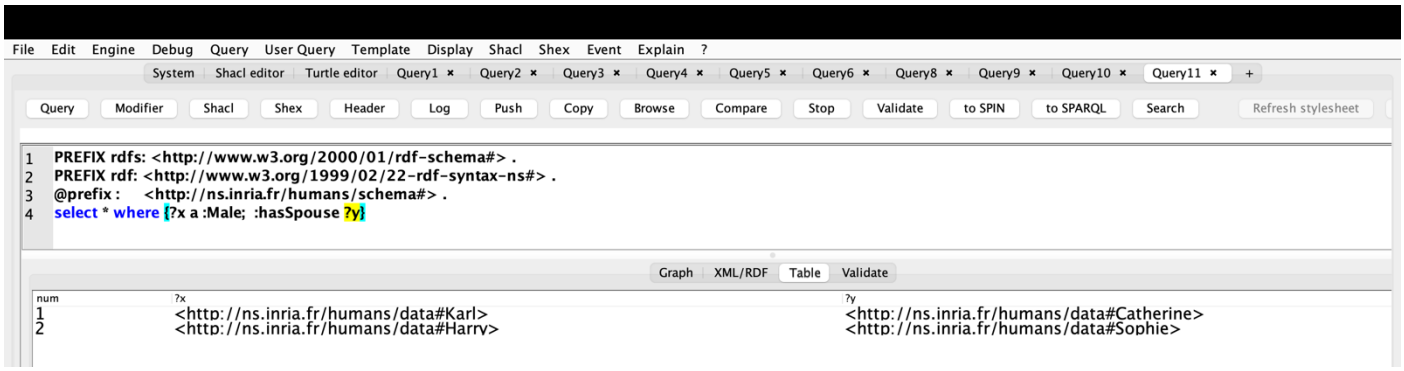
Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://ns.inria.fr/humans/schema#> .
select * where {?x a :Male;  :hasSpouse ?y}
```

Number of results and explanation:

We got one answer. Harry is a :Man and :Man is a subClassOf :Male so harry is also a :Male. It is the only :Man in the dataset who has the property hasSpouse. This inference has been possible thanks to the schema that we introduced earlier.

2. In the data declare that Lucas has father Karl. Reset Corese, reload the ontology and the data, and then rerun the query to find Males and their wives. Explain the new result.



Line added in RDF:

```
d:Lucas a :Man ;
   :age 12 ;
   :hasMother d:Catherine ;
   :hasFather d:Karl;
   :name "Lucas" ;
   :shirtsize 8 ;
   :shoesize 7 ;
   :trouserssize 28 .
```

Number of results before and after and explanation:

Before we got only one result but after we got two

Explanation: When we look at the property hasFather we see that it has a rds:range which is :Male. So by declaring that Lucas has for father Karl, the system infer that Karl is a :Male. That's why he is added to the new result.

```
:hasFather a rdf:Property ;
   rdfs:label "has for father"@en, "a pour père"@fr ;
   rdfs:comment "to have for parent a male."@en,
                "avoir pour parent un mâle."@fr ;
   rdfs:range :Male ;
   rdfs:subPropertyOf :hasParent .
```

Question 3

1. Write a query to find the Lecturers and their types. How many answers do you get? See how this typing is declared in the data and explain the result.

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX :     <http://ns.inria.fr/humans/schema#> .
select *
 where {?x a :Lecturer .
?x a ?y}
```

Number of results and your explanation:

We got 7 results for the person that are lecturer. Lecturer is a subClassOf of Person. So the system infer that Eve is also a Person.

It is clearly specified in the triples concerning Laura that she is a Person, Lecturer and a researcher. By analyzing the triples concerning Laura we see that:

```
d:Catherine a :Woman ;
   :hasMother d:Laura .
```

- Catherine has for mother Laura and :hasMother has range :female. So, the system infer that Laura is a female.
- Female is a subClassOf Animal, so the system also infer that Laura is of class animal.

2. Write a query to find common resources both of type Person and of type Male (instances of both classes). See how this typing is declared in the data and explain the presence of Jack.

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX :     <http://ns.inria.fr/humans/schema#> .
select * where {?x a :Person ;  a :Male}
```

Your explanation of the result:

```
d:Mark a :Person ;
    :age 14 ;
    :hasFather d:John ;
    :name "Mark" ;
    :shirtsize 9 ;
    :shoesize 8 ;
    :trouserssize 36 .
```

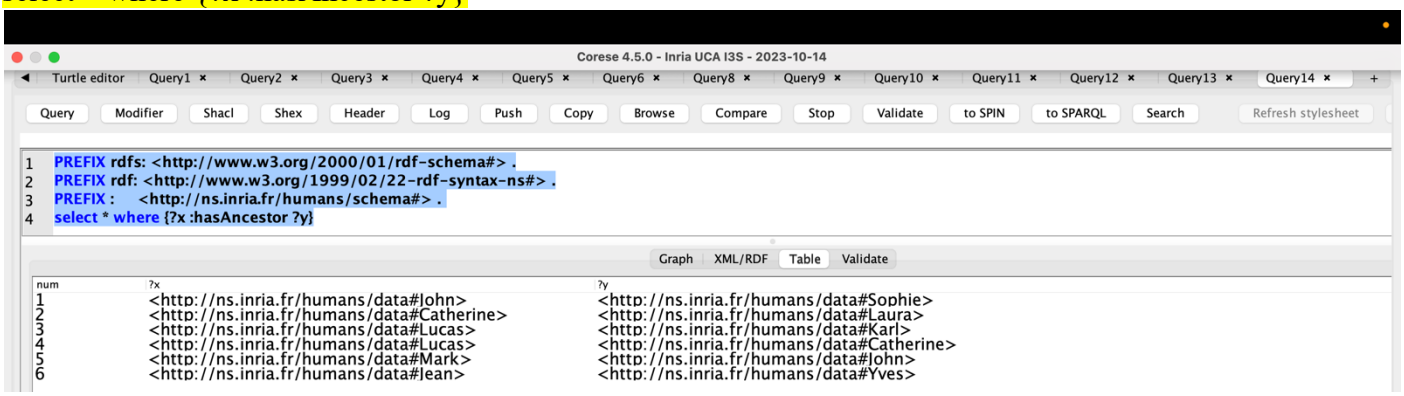
Mark has for father d:John and hasFather has for range Male. So the system infer that John is a Male.

Question 4

Write a query to find the hasAncestor relations. Explain the result after checking where this property is used in the data.

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX :     <http://ns.inria.fr/humans/schema#> .
select * where {?x :hasAncestor ?y}
```



Your explanation of the result:

The property hasParent is a sub Property of hasAncestor, and hasFather and hasMother are sub Properties of hasParent. so the system infer the results those results.

Question 5

1. Write a query to find the family cores (couples and their children) using a SELECT

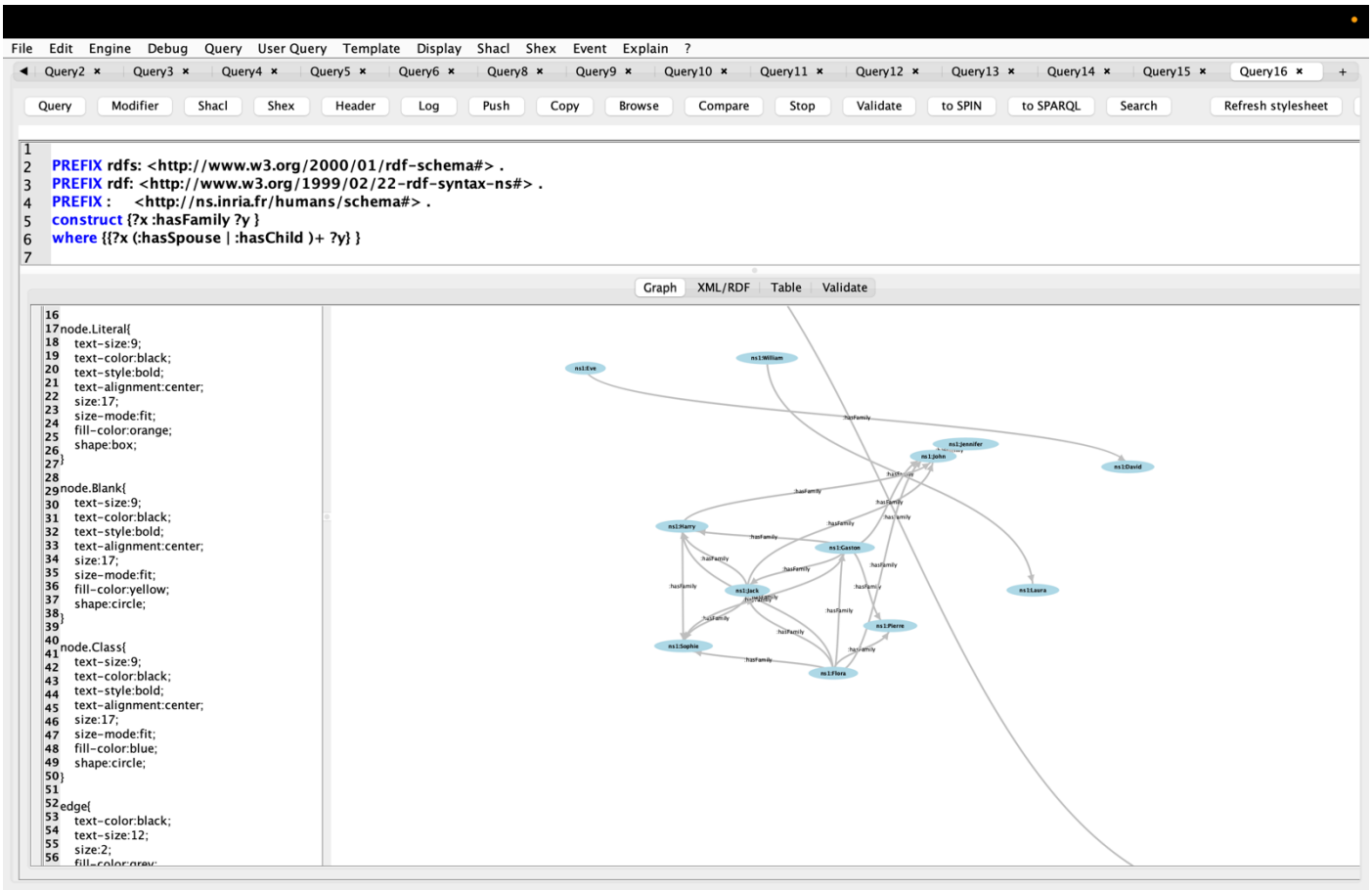
Query:

```
select *
where {
{?x (:hasSpouse | :hasChild )+ ?y}
}
```

2. Modify it to display the result with a CONSTRUCT query

Query:

```
construct {?x :hasFamily ?y }

where {{?x (:hasSpouse | :hasChild )+ ?y} }
```



Question 6

1. Declare the olderThan relationship in the schema to indicate between two persons which is eldest and construct the arcs between persons with a SPARQL query

Addition to schema:

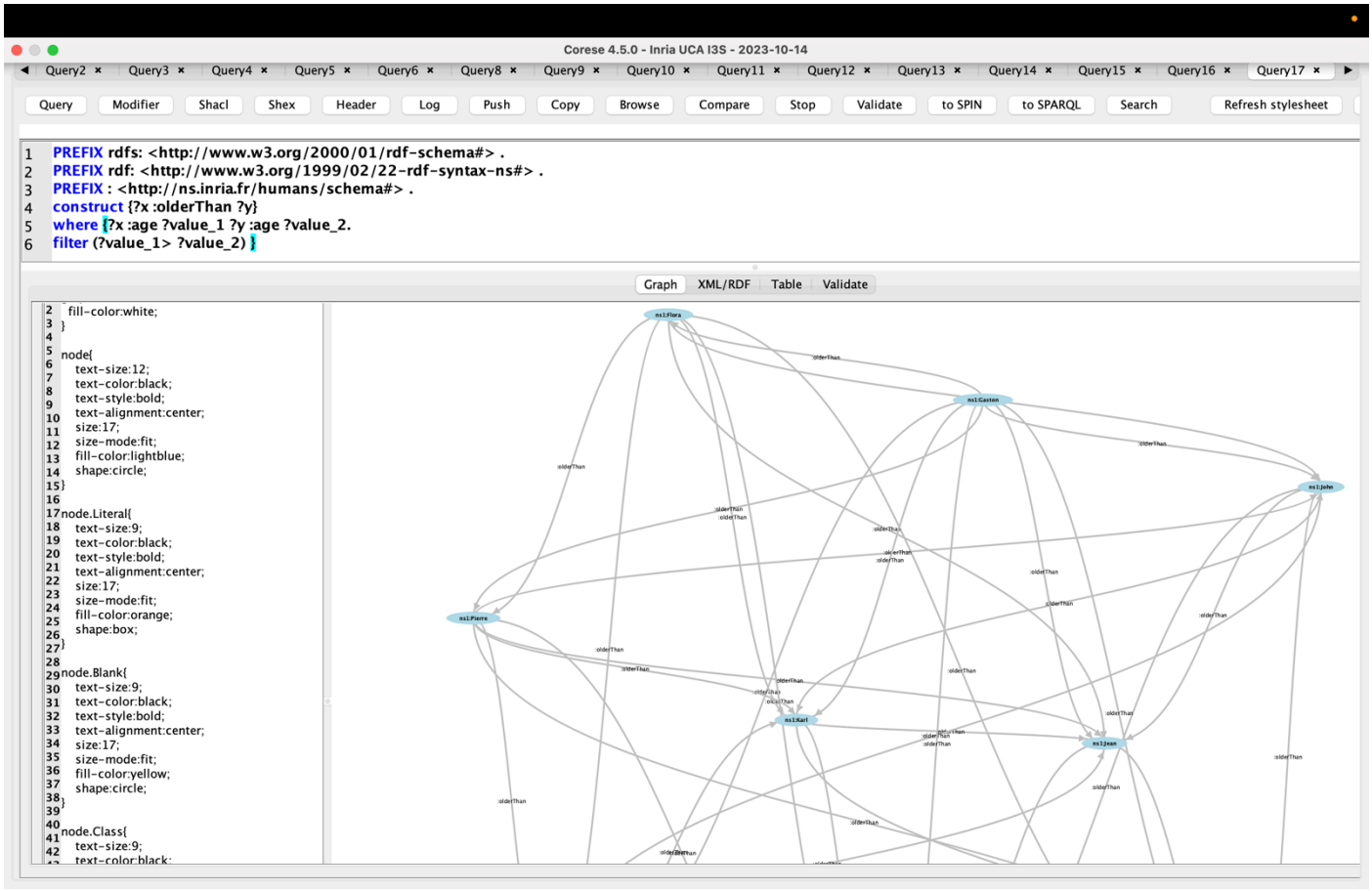
```
:olderThan a rdf:Property ;
        rdfs:domain h:Person ;
        rdfs:range h:Person .
```

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX: <http://ns.inria.fr/humans/schema#> .
construct {?x :olderThan ?y}
where {?x :age ?value_1 ?y :age ?value_2.
filter (?value_1> ?value_2) }
```

2. Find a query that generates only the minimum number of links without redundancy with olderThan transitivity.

Query:



Question 7

Write a query to find for John the properties which label contains the string "size" and the value of these properties.

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
PREFIX d: <http://ns.inria.fr/humans/data#> .
PREFIX :     <http://ns.inria.fr/humans/schema#> .

select *
where {d:John ?p ?v .
?p rdfs:label ?val .
 filter (contains(?val, "size"))
}
```

Question 8

Use the ontology to document your answers in natural language: write a query to find the types and properties of Laura in French.

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX d: <http://ns.inria.fr/humans/data#> .
PREFIX :     <http://ns.inria.fr/humans/schema#> .

select ?property ?y ?p ?v
where {d:Laura ?property ?y.
?property ?p ?v .
 filter (lang(?v)="fr") }
```

Create your own schema Family schema (can be done after the OWL practical session too if you are running out of time)

- Write the RDF schema that you used in the description of Jen in a RDFS Turtle (or in RDF/XML and then translate it) and save the RDFS Turtle in a file called “Family_schema.ttl” (or “Family_schema.xml”). Of course, this assumes that the URIs for the classes and properties declared/used must match in both files. You may have to update the files Jen.rdf and Jen.ttl to use your ontology.

Your schema:

##Schema initial

```
@prefix voca: <http://www.life.fr/voc#> .
@prefix : <http://www.dsti.fr/#> . # URI
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

voca:Man a rdfs:Class .
voca:Woman a rdfs:Class .
voca:name a rdf:Property ; rdfs:subPropertyOf rdfs:Literal .
voca:age a rdf:Property ; rdfs:subPropertyOf rdfs:Literal .
voca:hasChild a rdf:Property .
voca:hasSpouse a rdf:Property .
voca:Engineer a rdfs:Class .
```

##Schema final

```
@prefix voca: <http://www.life.fr/voca#> .
@prefix : <http://www.dsti.fr/#> . # URI
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

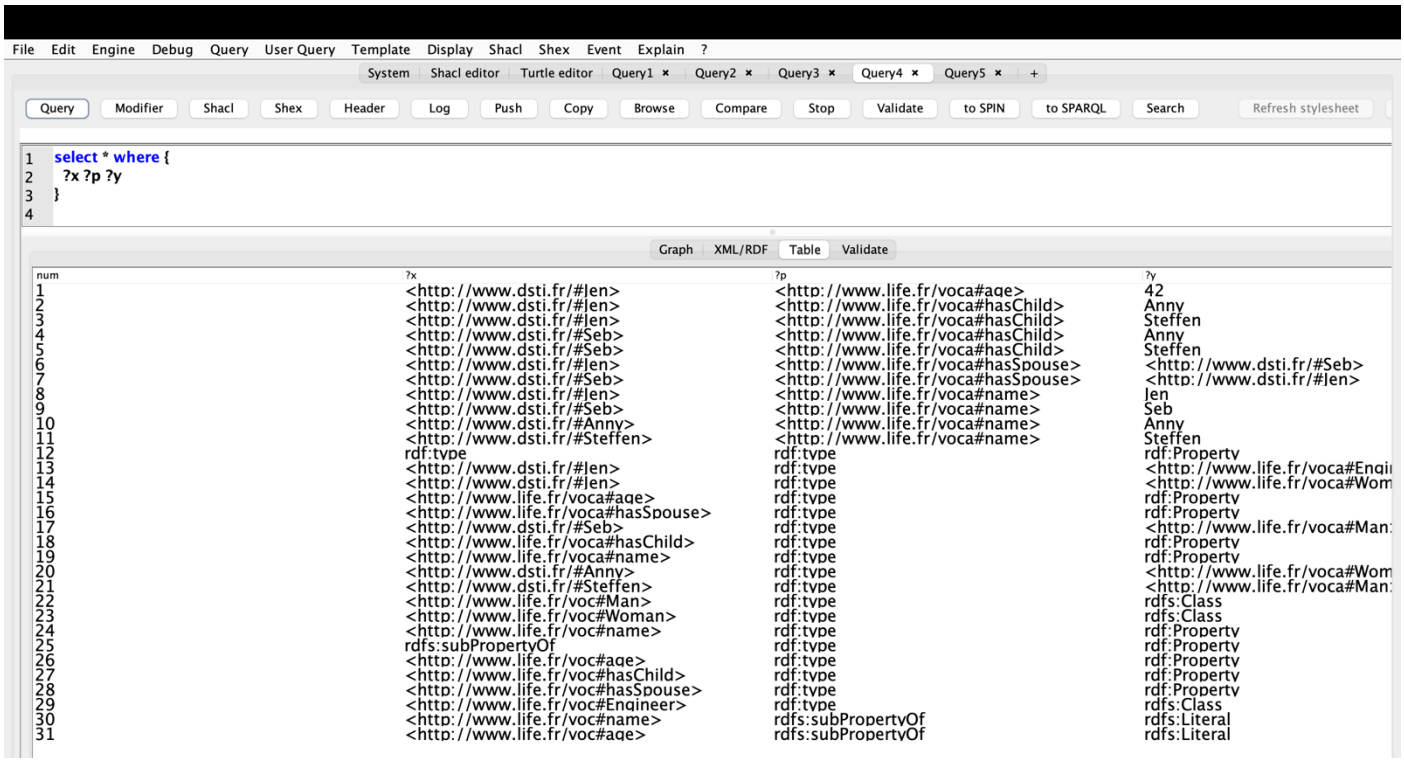
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

voca:Human a rdfs:Class .
voca:Man a rdfs:Class; rdfs:subClassOf voca:Human .
voca:Woman a rdfs:Class; rdfs:subClassOf voca:Human .
voca:name a rdf:Property ; rdfs:subPropertyOf rdfs:Literal .
voca:Engineer a rdfs:Class .
voca:age a rdf:Property ; rdfs:subPropertyOf rdfs:Literal .
voca:FamilyMember a rdfs:Class .
voca:familyLink a rdf:Property ;rdfs:domain voca:FamilyMember;
rdfs:range voca:FamilyMember .

voca:hasChild a rdf:Property ; rdfs:subPropertyOf voca:familyLink .
voca:hasSpouse a rdf:Property; rdfs:subPropertyOf voca:familyLink.

- Check that your RDF schema and RDF files are valid using the W3C's RDF validation service or other converter/ translators services.
- Launch the standalone interface of Corese and load your files Family_schema.ttl and Jen.ttl
- The interface contains a default SPARQL query:
  `Select ?x ?t where {?x rdf:type ?t}`
  Launch the query and look at the results.

Screenshot:



- Modify your ontology to declare the classes of Man and Woman as sub classes of Human (don't change the data), reload the schemas and data and search for the humans to see the results

Screenshot:

Explanation:

Given that Woman and Man are sub class of Human, every resource that is a man or a woman the system infer that it is also a Human.

- Modify your ontology to declare the properties hasChild and hasSpouse as sub properties of familyLink (don't change the data), reload the schemas and data and search for the family links to see the results.

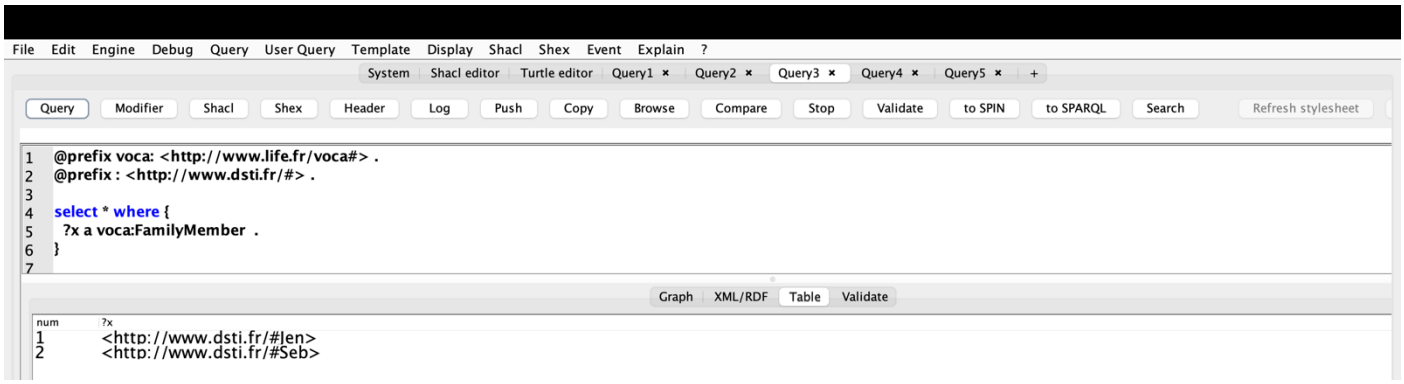
Screenshot:

Explanation:

hasChild and hasSpouse are subproperty of familyLink, so every resource that have an edge haschild and hadSpouse will also have an edfe familyLink, thus the results.

- Modify your ontology to declare the class FamilyMember and use it to specify the signature of the property familyLink (don't change the data) then reload the schemas and data and search for the family members.

Screenshot:



Explanation:

Jen and Seb are the only one in the data that are defined by a proprety that is a sub property of familyLink. So familylink being a sub property of family member , the the system infer that Jen and Seb belong to the class family member.

Lab session on OWL.

Software requirements

- The RDF XML online validation service by W3C: https://www.w3.org/RDF/Validator/
- The RDF online translator: http://rdf-translator.appspot.com/
- The SPARQL Corese engine (Corese-GUI jar file): https://project.inria.fr/corese/

A, Query data augmented by an OWL schema

Make a copy of the humans_schema.ttl file, name it humans_owl_schema.ttl and use it for the rest of the session. For each of the following statements, specify a SPARQL query that shows that the difference before and after running the OWL inferences: you will find that answers to these queries are different depending on whether you load the ontology humans_schema.ttl or the humans_owl_schema.ttl you modified.

Typically, to see the difference, you need to load the data (human_data), then make a query to check what was loaded and then load the schema (human_schema) and run the inference engine before making the same query again to witness changes.

Important: The "Engine Menu" allows you to control and witness the result of the inferences. If nothing is selected and if you run no rules you will just have the graph you loaded. If you start applying RDFS or OWL you will see new inferred results being added. For this practical session make sure to apply (unselect and reselect) "OWL RL" in the engine menu of Corese to run the rules to see the addition of results. Depending on the version of the CORESE-GUI you use you may have to repeat this operation several times to see all results.

1. Declare that `hasSpouse` is a symmetrical property and do the same for `hasFriend` .

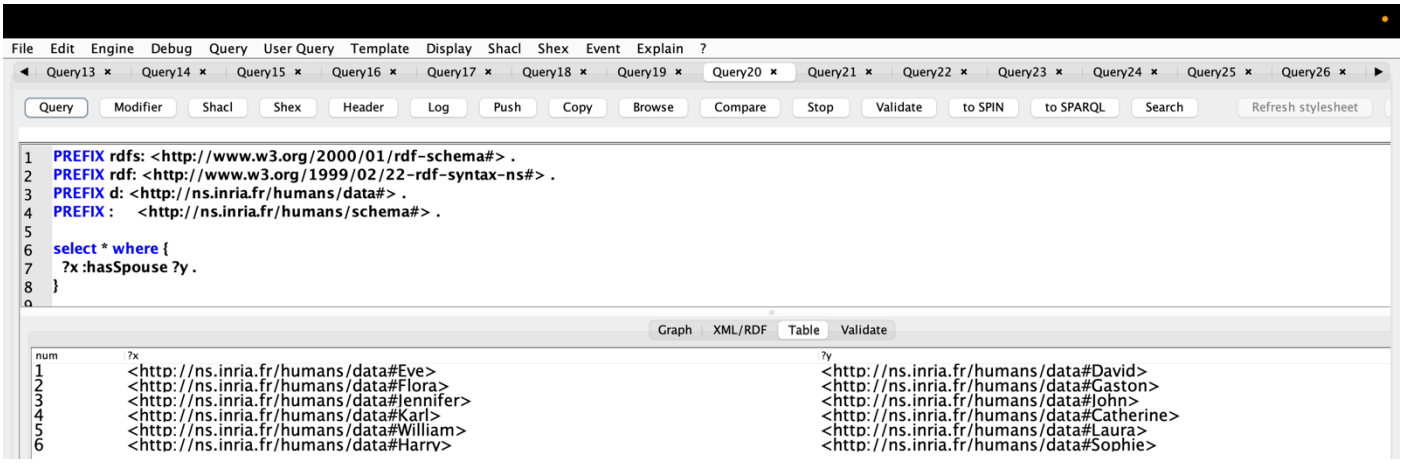
Code added to the schema:

```
owl:hasSpouse a owl:SymmetricProperty .
owl:hasFriend a owl:SymmetricProperty .
```

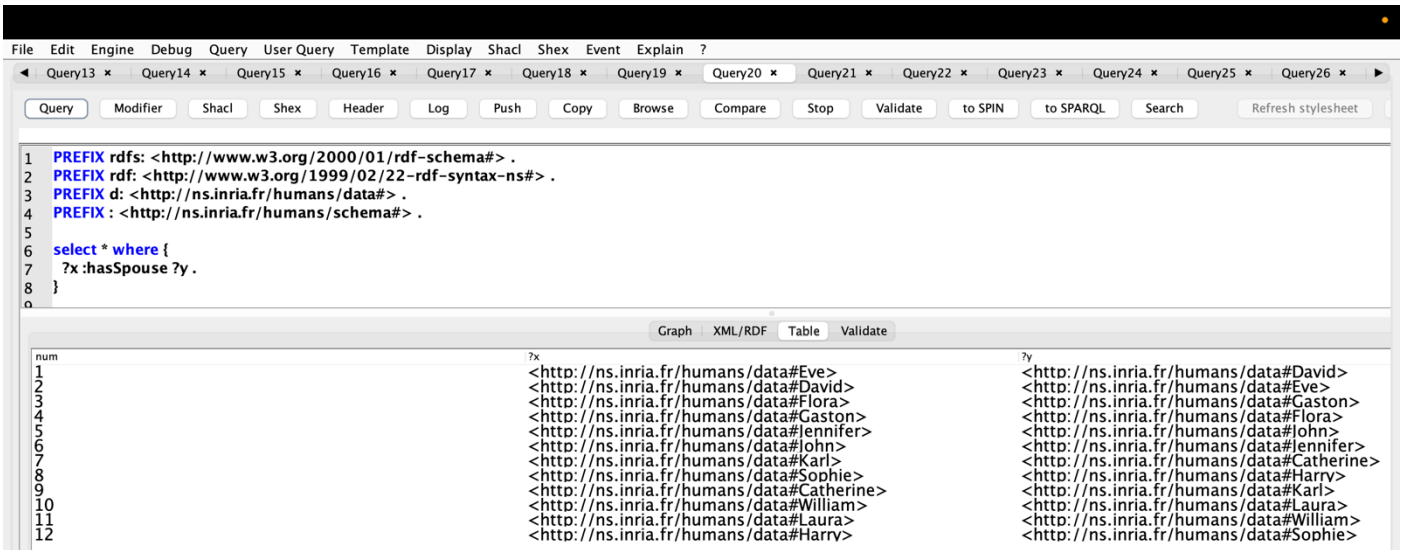
Query:

```
PREFIX :     <http://ns.inria.fr/humans/schema#> .
select * where {
  ?x :hasSpouse ?y .
}
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

Given that we define hasSpouse as symmetric, every time we declare that a resource ?x has for spouse a resource ?y, the system add the symmetric triple which is the resource ?y has for spouse the resource ?x. so the results are doubled.

2. Declare that `hasChild` is the inverse property of the `hasParent` property.

Code added to the schema:

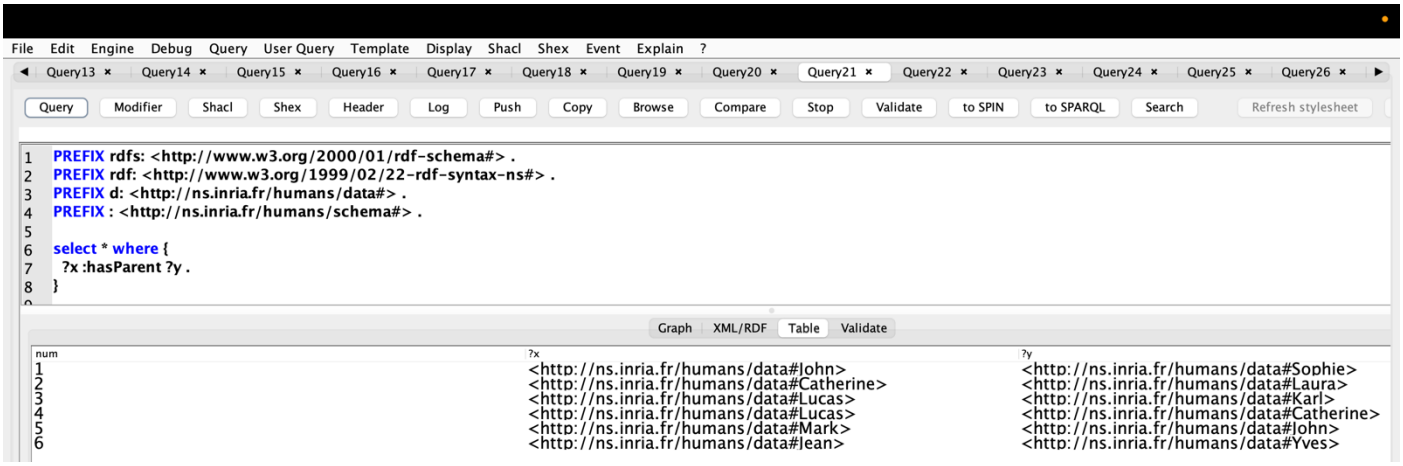
```
:hasChild owl:inverseOf h:hasParent .
```

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .

select * where {
  ?x :hasParent ?y .
}
```

Result before addition to the schema:

Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

For all the person that have the property ?x hasChild ?y , the system create ?y hasParent ?x because there are inverse, which allow more inferences on the data.

3. Declare `hasAncestor` as transitive property.

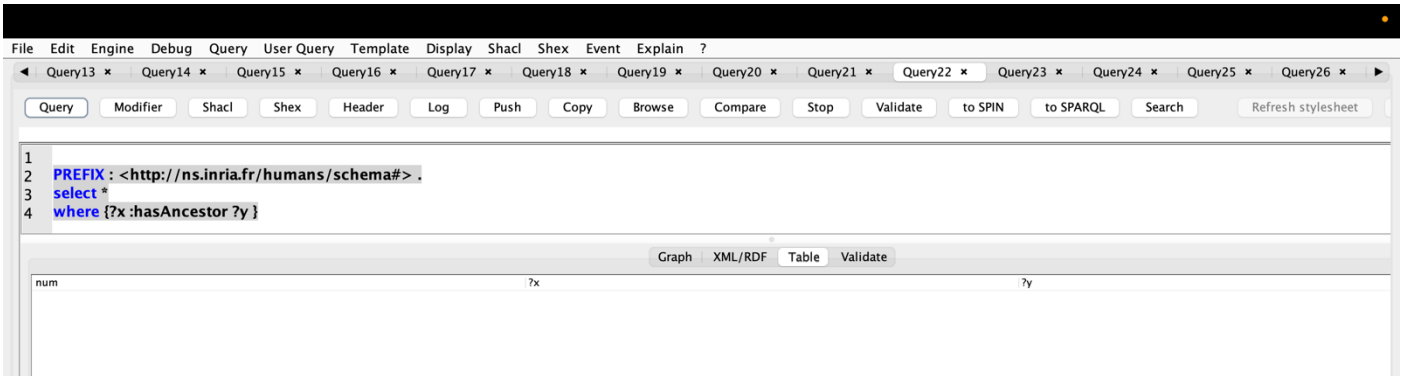
Code added to the schema:

```
:hasAncestor a owl:TransitiveProperty .
```

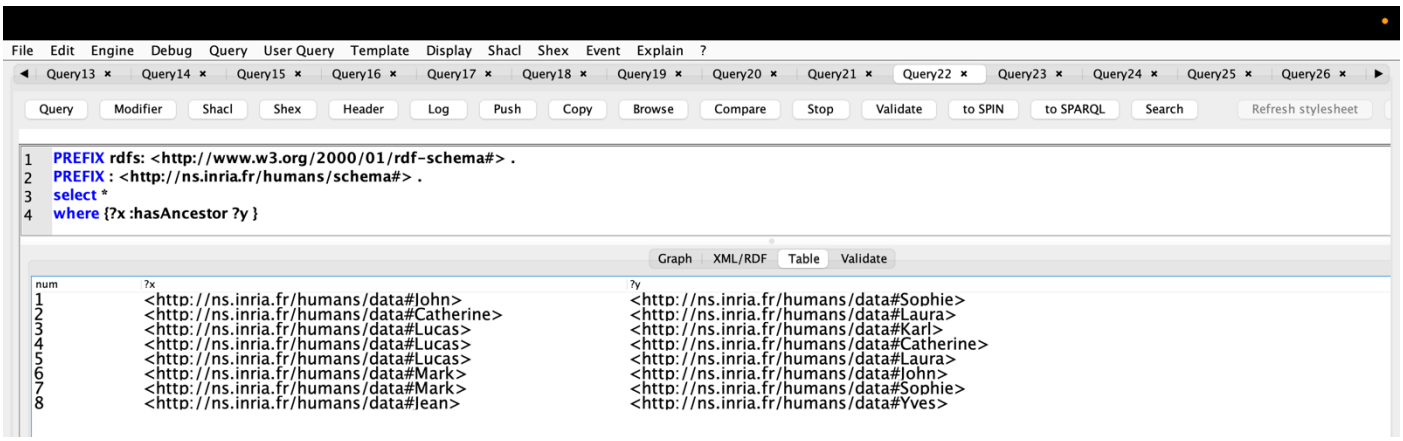
Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .
select *
where {?x :hasAncestor ?y }
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

hasParent is a sub property of hasAncestor, so any resources that have the property hasParent (?x hasParent ?y), the system will create (?x hasAncestor ?y) and if ?y hasParent ?z then the system will also create ?x hasAncestor ?z. this is why we get that augmented results.

4. Declare and define the chain property `hasSibling` has a super-property of the existing properties `hasBrother` and `hasSister`.

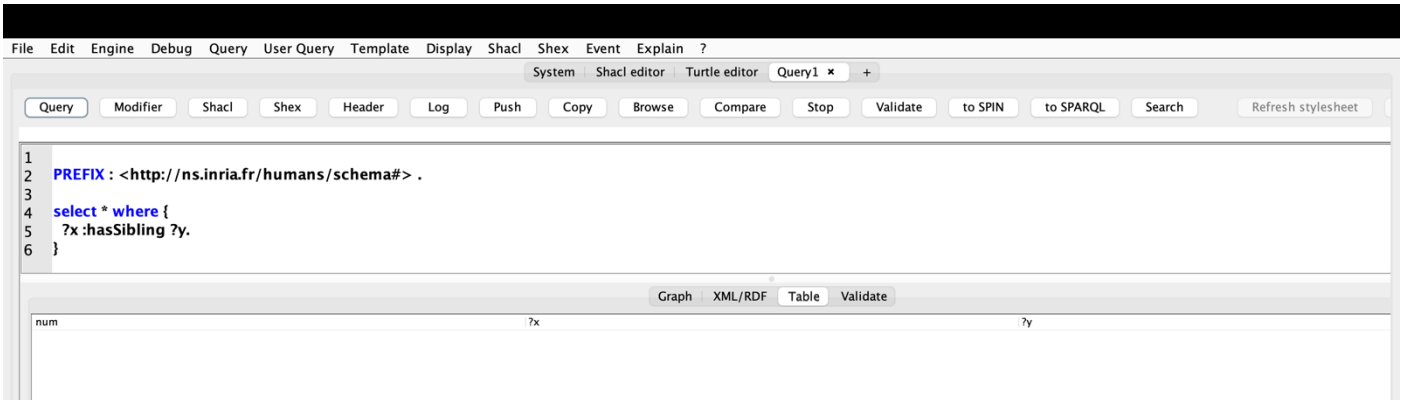
Code added to the schema:

```
:hasBrother a rdf:Property ;
            rdfs:subPropertyOf :hasSibling .

:hasSister a rdf:Property ;
            rdfs:subPropertyOf :hasSibling .
```

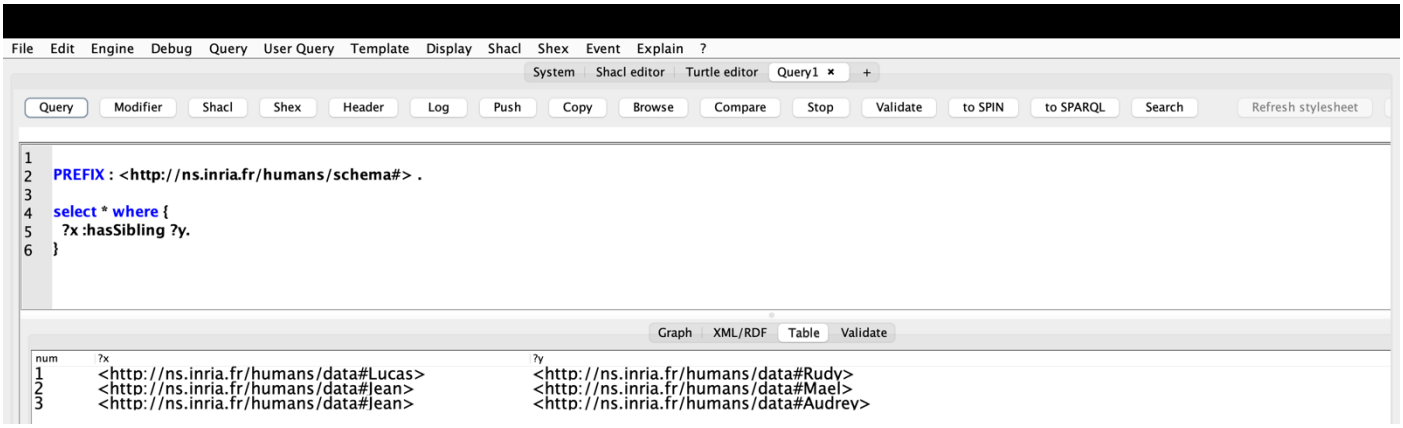
Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .

select *
where {?x :hasSibling ?y .}
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

For each person having a sister or a brother, the system infer that there are sibling

5. Declare and define the chain properties: `hasUncle` and `hasAunt` and in the data declare that Jack and Pierre are brothers and vice-versa.

Code added to the schema:

```
d:Jack a :Man ;
   :hasChild d:Harry ;
   :hasFriend d:Alice ;
   :name "Jack" ;
   :hasBrother d:Pierre .

d:Pierre a :Man ;
   :age 71 ;
   :name "Pierre" ;
   :shirtsize 9 ;
   :shoesize 8 ;
   :hasBrother d:Jack;
   :trouserssize 30 .

d:Harry a :Man ;
   :hasChild d:John ;
   :hasSpouse d:Sophie ;
   :hasParent d:Jack; # In order to see the difference otherwise I got nothing as a result.
   :name "Harry" .
```

In the schema I added:

```
:hasUncle owl:propertyChainAxiom ( :hasParent :hasBrother ) .
:hasAunt owl:propertyChainAxiom ( :hasParent :hasSister ) .
```

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .

select * where {
  ?x :hasUncle ?y .
}
```

Result before addition to the schema (reload then unselect and reselect “OWL RL”):

```
PREFIX : <http://ns.inria.fr/humans/schema#> .

select * where {
  ?x :hasUncle ?y .
}
```

Result after addition to the schema:

6. Declare the disjunction between `Male` and `Female`. Violate the constraint in the data, check the results and then remove the violation you created.

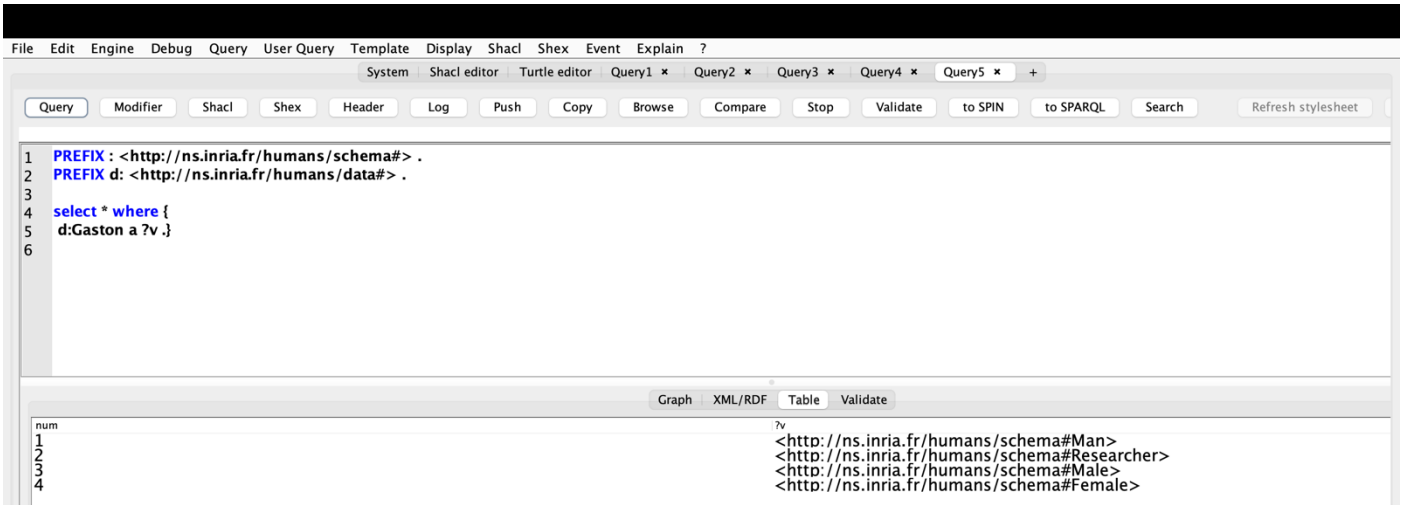
Code added to the schema:

```
:Male a rdfs:Class ;
   rdfs:label "male"@en, "mâle"@fr ;
   rdfs:comment "an animal that produces gametes (spermatozoa) that can fertilize female gametes (ova)."@en,
                "individu appartenant au sexe qui possède le pouvoir de fécondation."@fr ;
   rdfs:subClassOf :Animal ;
   owl:disjointWith :Female .
```

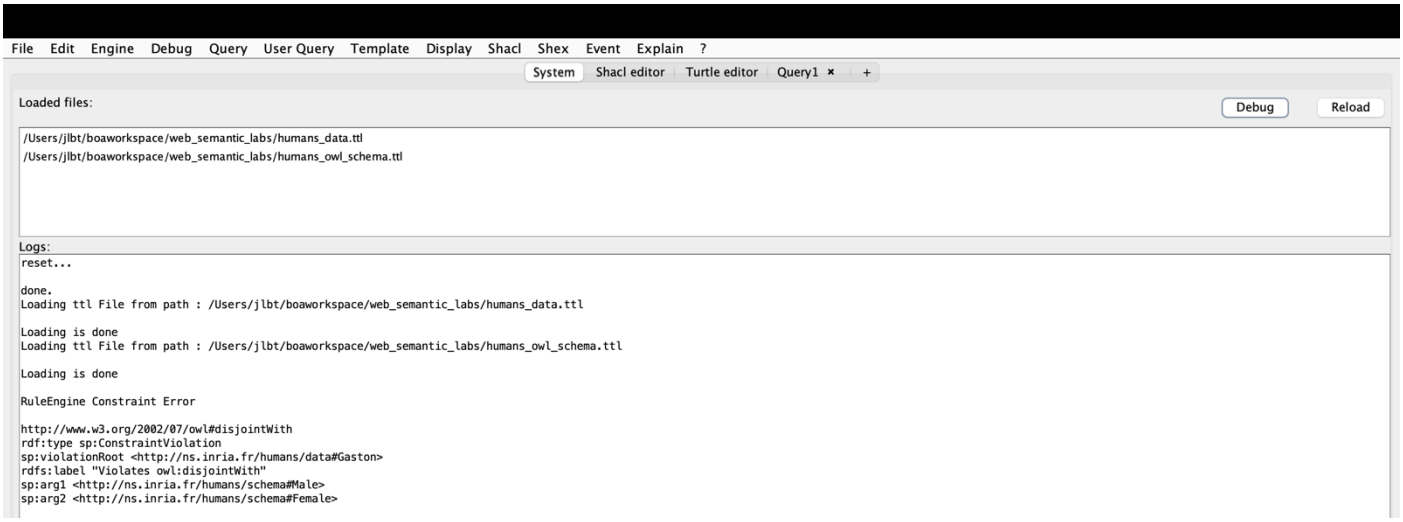
Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .
PREFIX d: <http://ns.inria.fr/humans/data#> .

select * where {
 d:Gaston a ?v .}
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

When we loaded the data for the first time, it was just a representation of the data in RDF format. So, there was no reasoning. The owl schema (that allow reasoning on data) allowed the system to find a incoherence in the data and to warn us about it.

7. Declare that the class `Professor` is the intersection of the class `Lecturer` and `Researcher` class.

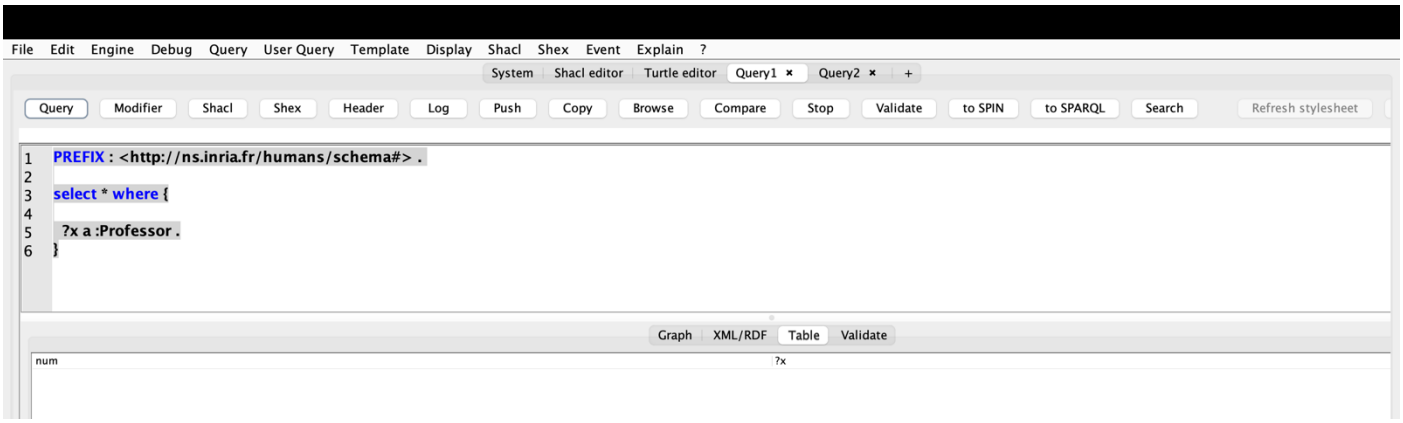
Code added to the schema:

```
:Professor a owl:Class;
owl:intersectionOf (:Researcher :Lecturer ) .
```

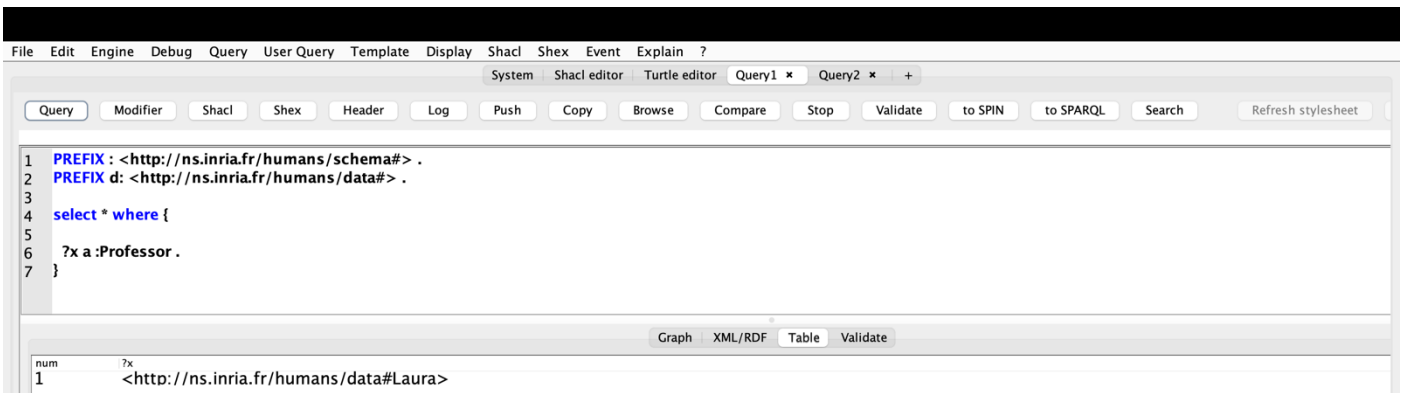
Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .

select * where {
  ?x a :Professor .
}
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

Each resource that are Researcher and Lecturer will be infer as being a Professor. Laura is the only one in that configuration in the database.

8. Declare that the `Academic` class is the union of classes `Lecturer` and `Researcher`.

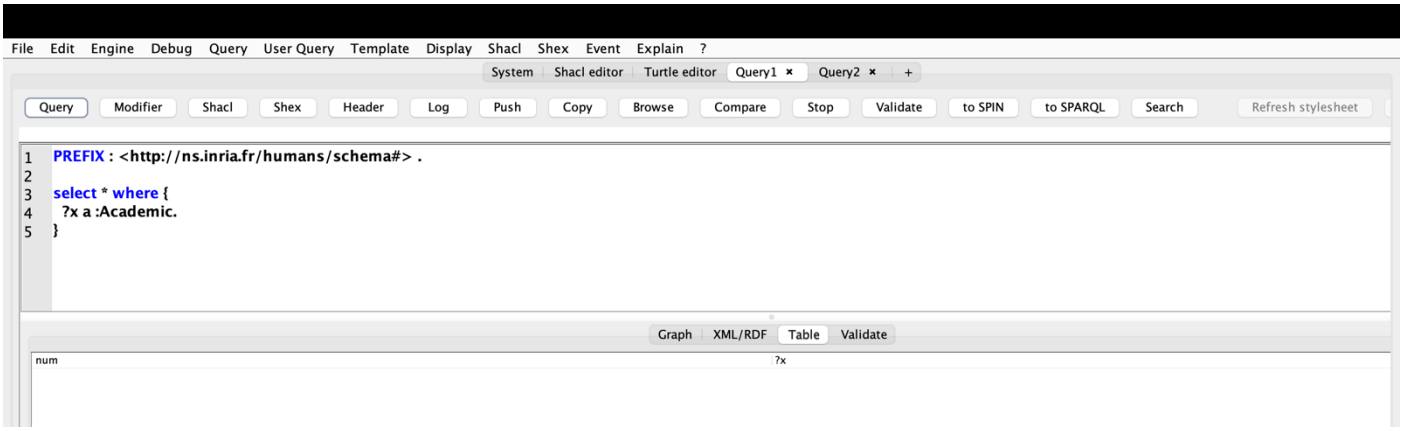
Code added to the schema:

```
:Academic a owl:Class ;
owl:unionOf (:Researcher :Lecturer ) .
```

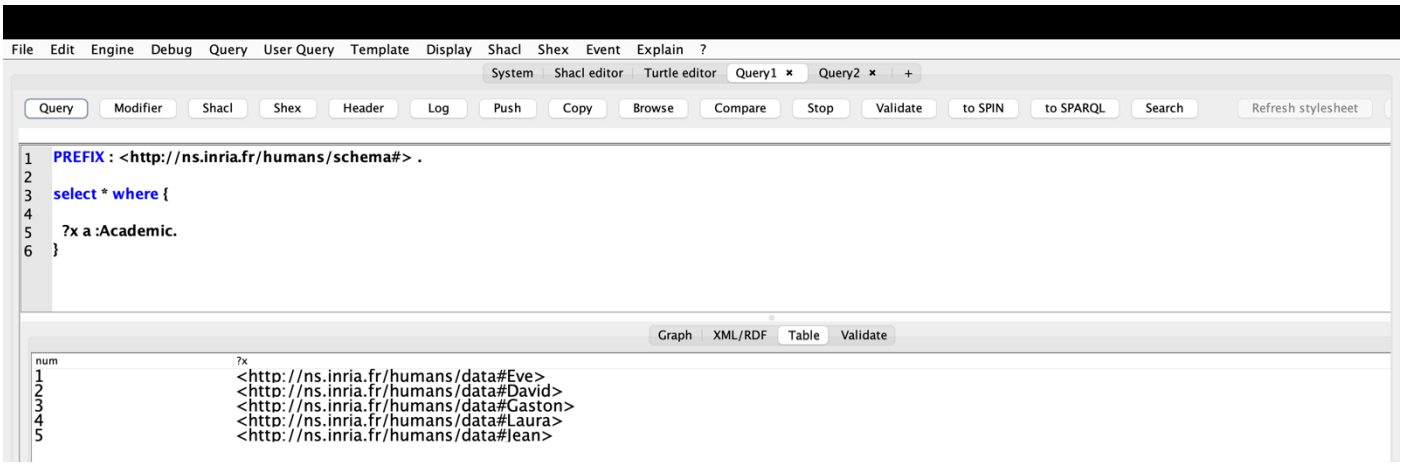
Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .

select * where {
  ?x a :Academic.
}
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

Each resource that are Researcher and/or Lecturer will be infer as being an Academic. Thus results.

9. Create a class `Organization` and its sub class `University`. Create a new property `mainEmployer`, with domain `Person` and range `Organization`. Use a restriction to declare that any `Professor` has for main employer a `University`.

Code added to the schema (new property, new classes and new restriction):

```
:Organization a rdfs:Class.
:University rdfs:subClassOf :Organization .

:mainEmployer rdf:property ;
              rdfs:domain :Person ;
              rdfs:range :Organization .


:Professor a owl:Class;
owl:intersectionOf (:Researcher :Lecturer );
```

```
rdfs:subClassOf [ a owl:Restriction ;
                  owl:onProperty :mainEmployer ;
                  owl:allValuesFrom :University ] .
```

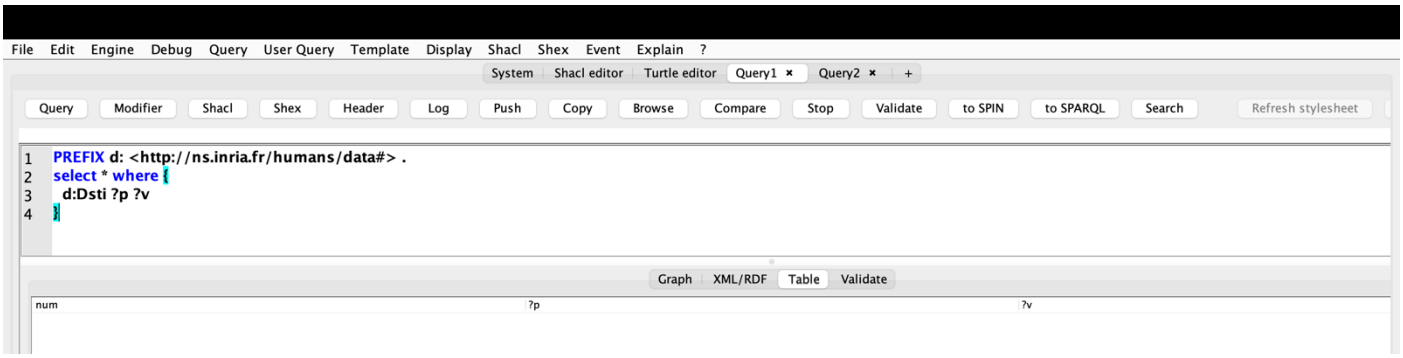
Code added to the data (just declare the main employer of a Professor):

```
d:Laura a :Lecturer, :Person, :Researcher ;
   :hasFriend d:Alice ;
   :name "Laura";
   :mainEmployer d:Dsti .
```

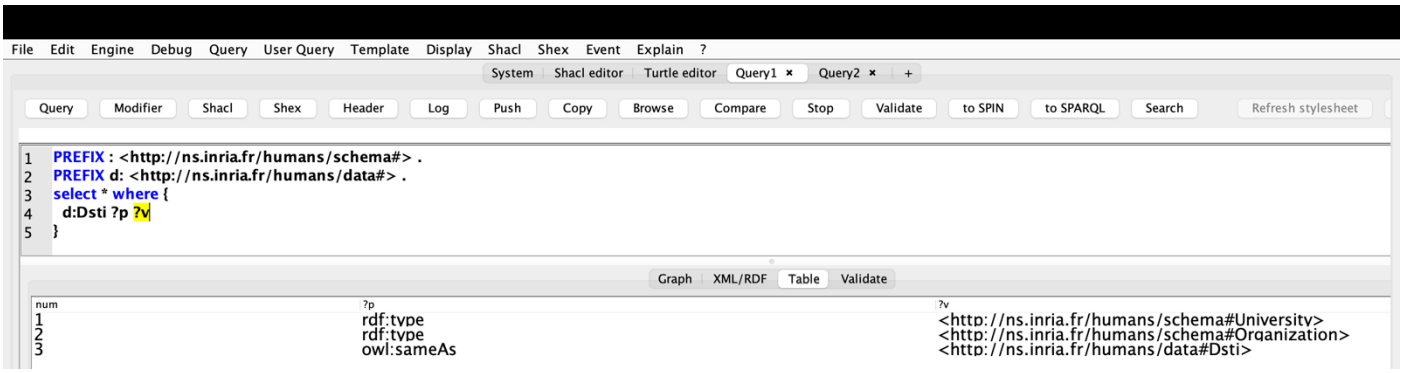
Query:

```
PREFIX d: <http://ns.inria.fr/humans/data#> .
select * where {
  d:Dsti ?p ?v
}
```

Result before addition to the schema:



Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

Dsti is a value of mainEmployer and mainEmployer has for range University, so Dsti is a university. Or University is a subclass of organization, so Dsti is also an Organisation. The system infer that Dsti is an Organization and a University it also infer that it the same as its URI which is True.

10. Use a restriction to declare that any person must have a parent who is a woman. For this last statement, you need to run the rule engine after loading the ontology and data.

Code added to the schema:

```
:Person a rdfs:Class ;
   rdfs:label "human"@en, "human being"@en, "person"@en,
              "homme"@fr, "humain"@fr, "personne"@fr, "être humain"@fr ;
   rdfs:comment "a member of the human species"@en,
```

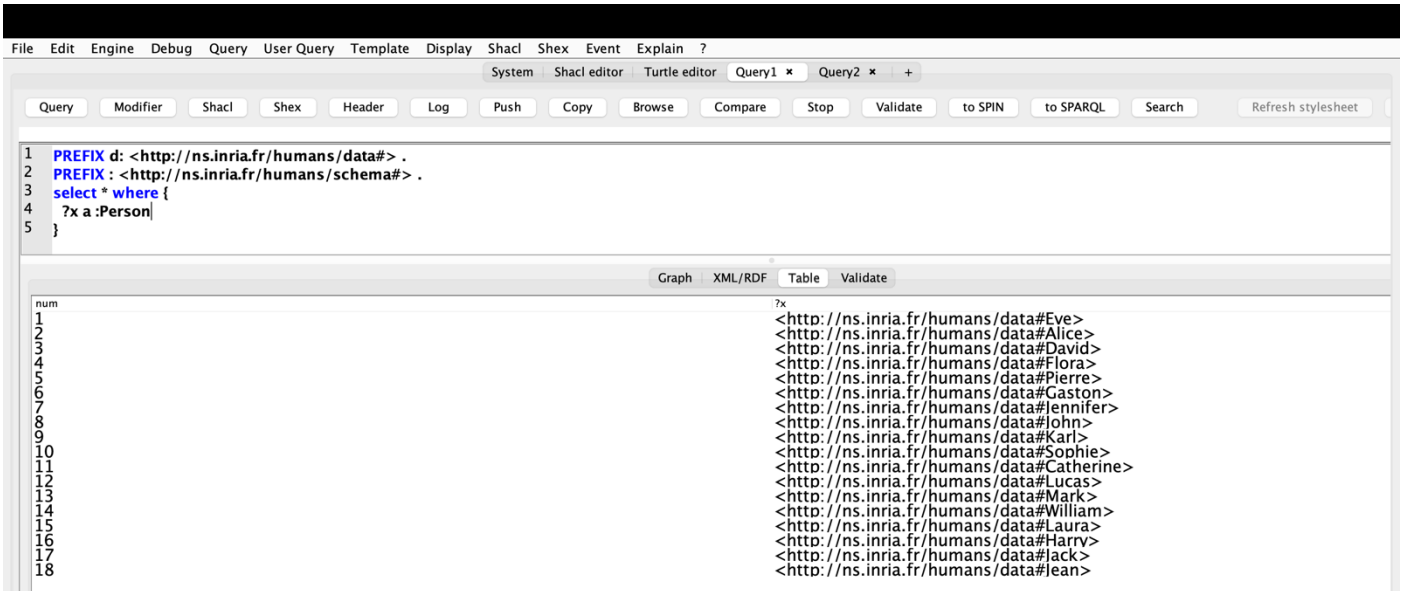
```
            "un membre de l'espèce humaine."@fr ;
    rdfs:subClassOf :Animal, [
            a owl:Restriction ;
            owl:onProperty :hasParent ;
            owl:someValuesFrom :Woman ] .
```

Query:

```
PREFIX : <http://ns.inria.fr/humans/schema#> .
select * where {
  ?x a :Person
}
```

Result before addition to the schema:

Result after addition to the schema (reload then unselect and reselect “OWL RL”):



Explanation:

All the person we obtained after adding the schema , has at least a either an instance of the class :Woman or a value of the data rang.

For example let’s take Gaston, he is a researcher and a researcher is a subclass of person so Gaston is an animal and also belongs to the blank node class defining the constraint on the property hasParent

B, Make your own OWL models:

For each one of the following OWL primitives imagine a definition that could use it and provide that definition in OWL using your preferred syntax (RDF/XML or N3/Turtle). For instance a possible definition

using owl:TransitiveProperty would be a definition of the Ancestor property. For each primitive in the following list you imagine the definition of a class or property that was not given in the course and you give that definition in English and in OWL.

1. owl:oneOf

   :CarColor rdf:type owl:Class ;
   owl:one of (:Blue :Green :Brown :Black) .
   CarColor is an owl class and instances of the class :CarColor can only take values from the specified list (:Blue :Green :Brown :Black)

2. owl:unionOf

   :PrimaryColor rdf:type owl:Class ;
   owl:unionOf (ex:Red ex:Green ex:Blue)
   :PrimaryColor are colors that belong to at least one of the classes :Red, :Green, or :Blue

3. owl:intersectionOf
   :Father rdf:type owl:Class ;
   owl:intersectionOf (:Male :Parent) .

   :Father represents the set of individuals who are both males and parents
4. owl:complementOf
   :EmptySet a owl:Class; owl:complementOf :FullSet

   Anything that is not in the set :FullSet belongs to the set :EmptySet.
5. owl:disjointWith
   or owl:AllDisjointClasses
   or owl:disjointUnionOf

   :Apple rdf:type owl:Class ;
   owl:disjointWith :Banana .

   If something is an apple, it cannot be a banana, and vice versa.

6. owl:ObjectProperty
   owl:hasKey a owl:ObjectProperty;
   :Car owl:hasKey;
   (:licensePlate :Color) .

   It defines relations between resources only. For example between :Car, :licensePlate and :Color

7. owl:DatatypeProperty
   Indicate that a literal value could be possibly typed.

   :hasHeight a owl:DatatypeProperty .
   :alice :hasHeight "170.0"^^xsd:double .

8. owl:SymmetricProperty
   or owl:AsymmetricProperty

:hasMeet a ow:Class ;
owl:SymmetricProperty .

If individual A has a meet with individual B, then individual B also has a meet with individual A.

9. owl:inverseOf
:Call a owl:Class;
Owl:InverseOf :HasBeenCalled .

If A has called B, then it implies that B has been called by A.

10. owl:TransitiveProperty
:liveWith a owl:Class;
owl:TransitiveProperty .

It suggests a transitive relationship where if individual A lives with individual B, and individual B lives with individual C, then it implies that individual A also lives with individual C.

11. owl:propertyDisjointWith
:hasParent owl:Class;
Owl:propertyDisjointWith :hasNoParent .

If an individual has a parent according to the property :hasParent, it cannot simultaneously have no parent according to the property :hasNoParent, and vice versa.

12. owl:ReflexiveProperty
or owl:IrreflexiveProperty
asOldAs a owl:Class;
owl:ReflexiveProperty .

Every individual is as old as itself

13. owl:propertyChainAxiom

:ElectricCar rdf:type owl:ObjectProperty ;
owl:propertyChainAxiom (:Car :Electric) .

Instances of :ElectricCar are inferred when an individual is both an instance of :Car and an instance of :Electric.

14. owl:FunctionalProperty
:birthdate a owl:FunctionalProperty .

It implies that each individual can have only one birthdate

15. owl:InverseFunctionalProperty
:telephone a owl:InverseFunctionalProperty .
If two individuals have the same telephone number, then they are considered to be the same.

16. owl:hasKey
:Car owl:hasKey
(:licensePlate :Color) .

Each instance of :Car must have a unique combination of values for the properties :licensePlate and :Color. It is impossible to find two cars having the same combination of license plate and color

17. owl:allValuesFrom

```
:Carnivore a owl:Class ;
rdfs:subClassOf :Animal,
        [ a owl:Restriction ;
        owl:onProperty :eats;
        owl:allValuesFrom :Meat ] .
```

:Carnivore are a subclass of :Animal that must eat only meat.

18. owl:someValuesFrom

```
:ChessPlayer a owl:Class ;
[ a owl:Restriction ;
owl:onProperty :hobby ;
owl:someValuesFrom :Chess ] .
```

:ChessPlayer must have at least one hobby that is playing chess.

19. owl:hasValue

```
:Car a owl:Class ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:onProperty :nbWheels ;
owl:hasValue "4" ] .
```

All instances of subclasses of ":Car" must have exactly four wheels.

20. owl:maxCardinality
    or owl:minCardinality

```
:Car a owl:Class ;
rdfs:subClassOf
[ a owl:Restriction ;
     owl:maxCardinality "1" ;
     owl:onProperty :licensePlate ] .
```

All instances of :Car can have at most one associated license plate.

21. owl:qualifiedCardinality

```
:Engine a owl:Class ;
rdfs:subClassOf [ a owl:Restriction ;
owl:onClass :Car ;
owl:onProperty :licensePlate ;
 owl:qualifiedCardinality "1" ] .
```

All subclasses of :Engine must be associated with exactly one car license plate.