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 do 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 is the maker of http://ns.inria.fr/fabien.gandon#me wich is also its primary topic

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.

The triple is:

```
dbr:Paris       dct:subject     dbc:Prefectures_in_France ,
                <http://dbpedia.org/resource/Category:Cities_in_\u00CEle-de-France> ,
                dbc:Catholic_pilgrimage_sites ,
                dbc:Cities_in_France ,
                <http://dbpedia.org/resource/Category:3rd-century_BC_establishments> ,
                dbc:Capitals_in_Europe ,
                dbc:European_culture ,
                dbc:Paris ,
                <http://dbpedia.org/resource/Category:Departments_of_\u00CEle-de-France> ,
                dbc:Gallia_Lugdunensis ,
                dbc:French_culture ,
                dbc:Companions_of_the_Liberation ,
                dbc:Populated_places_established_in_the_3rd_century_BC .
```

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

102

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.

   curl -o human-data.xml http://ns.inria.fr/humans/humans_data.xml

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"

curl -o human_data.ttl http://ns.inria.fr/humans/humans_data.ttl

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

   http://ns.inria.fr/humans/data# wich the prefix name is d

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?

@prefix d: <http://ns.inria.fr/humans/data#> .

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 default one declared with @prefix : <http://ns.inria.fr/humans/schema#> .

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

John is a person of 37 years old. He has a parent named Sophie, its shirtsize is 12, shoesize 14, trousersize 44. He is friend with Alice. His spouse is Jennifer. Mark is his children and Harry is his father.

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 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://www.unice.fr/voc#> .
@prefix : <http://www.unice.fr/data#> .
:Jen
  a voc:Woman ;
  voc:name "Jen" ;
  voc:age 42 ;
  voc:hasSpouse :Seb ;
  voc:hasChild :Steffen, :Anny ;
  rdf:Type voc:Engineer .

:Anny
  voc:name "Anny" ;
  a voc:Woman .

:Seb
  a voc:Man ;
  voc:name "Seb" ;
  voc:hasChild :Steffen, :Anny .

:Steffen
  a voc:Man ;
  voc: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:voc="http://www.unice.fr/voc#">

  <rdf:Description rdf:about="http://www.unice.fr/data#Jen">
    <rdf:type rdf:resource="http://www.unice.fr/voc#Woman"/>
    <voc:name>Jen</voc:name>
    <voc:age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42</voc:age>
    <voc:hasSpouse>
      <voc:Man rdf:about="http://www.unice.fr/data#Seb">
        <voc:name>Seb</voc:name>
        <voc:hasChild rdf:resource="http://www.unice.fr/data#Steffen"/>
        <voc:hasChild rdf:resource="http://www.unice.fr/data#Anny"/>
      </voc:Man>
    </voc:hasSpouse>

    <voc:hasChild rdf:resource="http://www.unice.fr/data#Steffen"/>
    <voc:hasChild rdf:resource="http://www.unice.fr/data#Anny"/>
    <rdf:Type rdf:resource="http://www.unice.fr/voc#Engineer"/>
  </rdf:Description>

  <voc:Woman rdf:about="http://www.unice.fr/data#Anny">
    <voc:name>Anny</voc:name>
  </voc:Woman>

  <voc:Man rdf:about="http://www.unice.fr/data#Steffen">
    <voc:name>Steffen</voc:name>
  </voc:Man>

</rdf:RDF>
```

The RDF/XML obtained by converting the turtle version is the same as mine although it has the prefix ns0 which is the default naming for generated prefix

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:

NodeShape

What is the target of that shape:

Person

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

The property name must have at least one value, if not the error with severity Violation must be raise and the message “A person must have a name” wil be return

What is the severity level of that constraint?

if the constraint is not respected, the error with severity Violation must be raise and the message “A person must have a name” wil be return

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:

The report says the node Karl of type Person doesn’t have a property name wich is a violation of the constraint so the dataset isn’t conform.

Add your constraints

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

```
:PersonShape a sh:NodeShape ;

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

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:

The report is saying that David and Eve nodes of type person should have a property age which is a warning and the node Karl of type Person doesn’t have a property name wich is a violation of the constraint so the dataset isn’t conform.

If the constraint on age is put after the one on name, only the message about name is obtained as it’s a violation constraint and the validation end when encountered.

Extend the shape to add a constraint of severity level “Info” enforcing that a person’s name should be in English:

```
]; (replacing “].”)
sh:property [
                sh:message "a Person’name should be in english"@en;
                sh:path :name ;
                sh:languageIn ( "en" ) ;
                sh:severity sh:Info ;
        ].
```

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:

The report inform us that the node David, Laura, Eve, William, John, Mark and Karl should have an English name i.e a a property name which the declared language is the english. the rest of the message is the same than the previous question

Lab session on SPARQL.

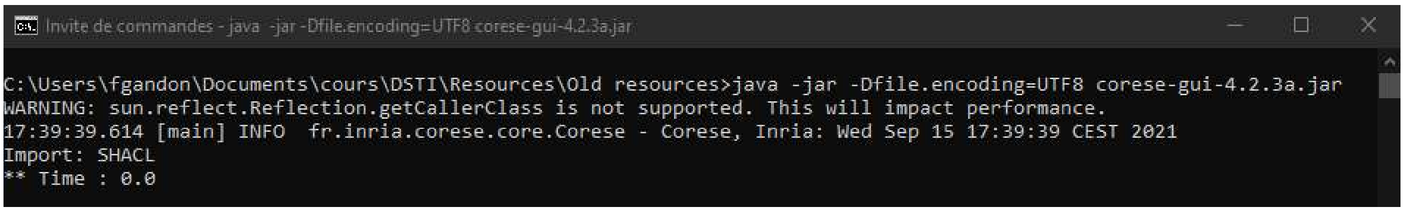
Software requirements

- 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

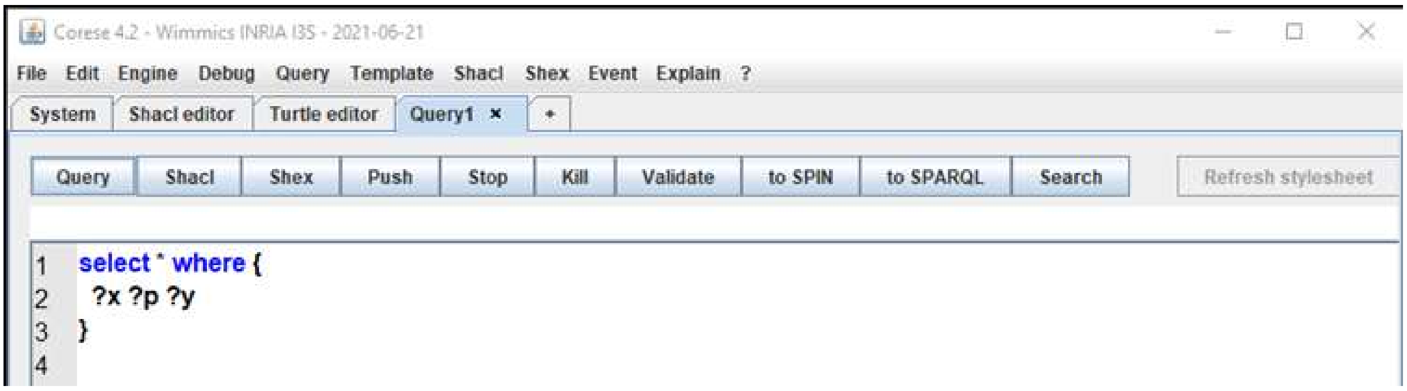
If you haven't done it yet download the SPARQL Corese engine.

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:



```
Invite de commandes - java  -jar -Dfile.encoding=UTF8 corese-gui-4.2.3a.jar

C:\Users\fgandon\Documents\cours\DSTI\Resources\Old resources>java -jar -Dfile.encoding=UTF8 corese-gui-4.2.3a.jar
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
17:39:39.614 [main] INFO  fr.inria.corese.core.Corese - Corese, Inria: Wed Sep 15 17:39:39 CEST 2021
Import: SHACL
** Time : 0.0
```

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.



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

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:

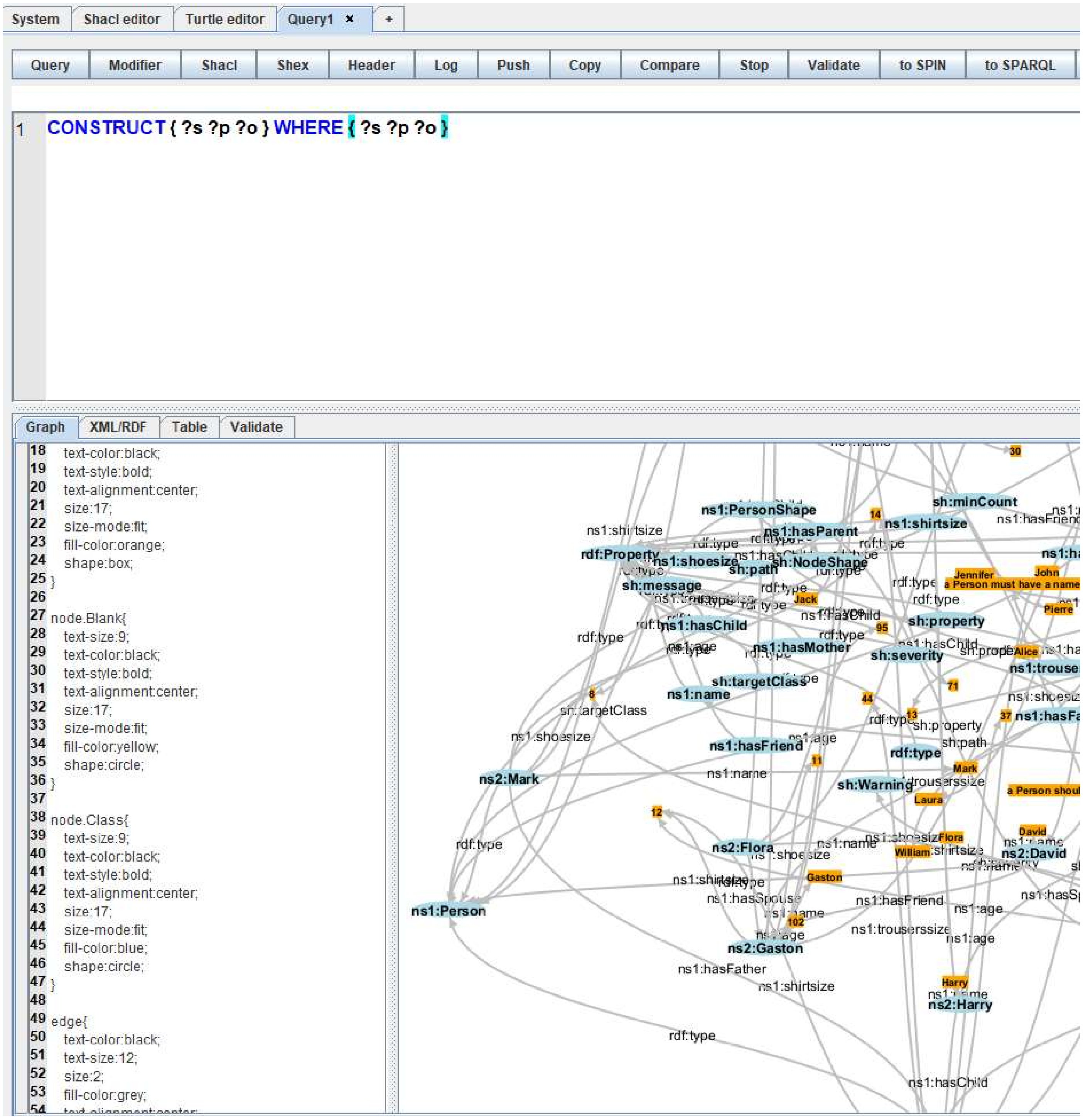
Create a new tab to enter the following query and explain what it does and the results you get. This is a good way to familiarize yourself with the data.

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

Explanation:

It returns a single RDF graph specified by a graph template, here all the graph

Screenshot:



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

```
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:orange;
    shape:box;
}

node.Blank{
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:yellow;
    shape:circle;
}

node.Class{
    text-size:9;
    text-color:black;
    text-style:bold;
    text-alignment:center;
    size:17;
    size-mode:fit;
    fill-color:blue;
    shape:circle;
}

edge{
    text-color:black;
    text-size:12;
    size:2;
    fill-color:grey;
```

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.

Select all nodes and their type whose type starts by http://ns.inria.fr/humans/schema#

Run this query. How many answers do you get?

21

Find John and his types in the answers.

John's types:

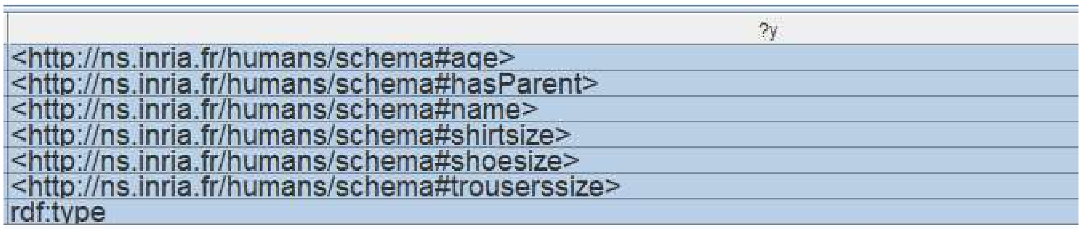
<http://ns.inria.fr/humans/schema#Person>

Question 3:

In the previous answer, locate the URI of John.

1. formulate a SELECT query to find all the properties of John, using his URI
   Query
   select ?y where { <http://ns.inria.fr/humans/data#John> ?y ?z . }

   Results:



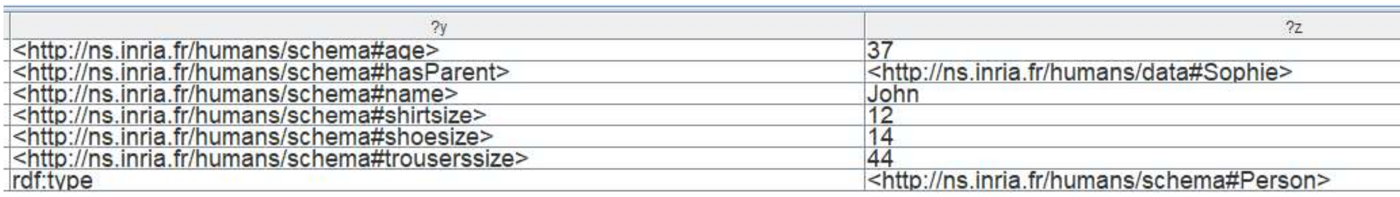
?y
<http://ns.inria.fr/humans/schema#age>
<http://ns.inria.fr/humans/schema#hasParent>
<http://ns.inria.fr/humans/schema#name>
<http://ns.inria.fr/humans/schema#shirtsize>
<http://ns.inria.fr/humans/schema#shoesize>
<http://ns.inria.fr/humans/schema#trouserssize>
rdf:type

2. request a description of John using the SPARQL clause for this.

Query

select ?y ?z where { <http://ns.inria.fr/humans/data#John> ?y ?z . }

Results:



?y	?z
<http://ns.inria.fr/humans/schema#age>	37
<http://ns.inria.fr/humans/schema#hasParent>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/schema#name>	John
<http://ns.inria.fr/humans/schema#shirtsize>	12
<http://ns.inria.fr/humans/schema#shoesize>	14
<http://ns.inria.fr/humans/schema#trouserssize>	44
rdf:type	<http://ns.inria.fr/humans/schema#Person>

Question 4

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.

Get all the node that have a spouse and the value of the spouse

Run this query. How many answers do you get?

6

Question 5:

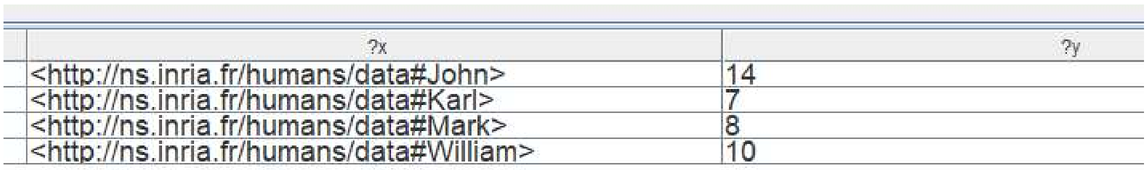
In the RDF file, find the name of the property that is used to give the shoe size of a person.

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

Query:

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

Result:



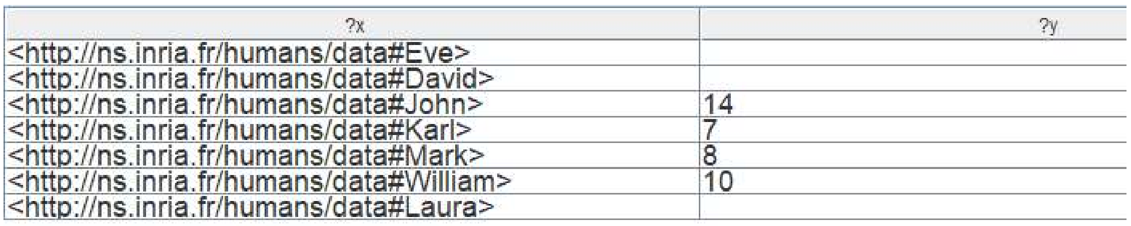
?x	?y
<http://ns.inria.fr/humans/data#John>	14
<http://ns.inria.fr/humans/data#Karl>	7
<http://ns.inria.fr/humans/data#Mark>	8
<http://ns.inria.fr/humans/data#William>	10

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

Query:

```
prefix h: <http://ns.inria.fr/humans/schema#>
select ?x ?y where { ?x a h:Person .
optional {?x h:shoesize ?y . } }
```

Result:



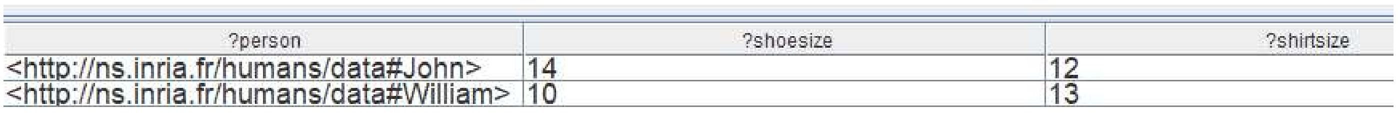
?x	?y
<http://ns.inria.fr/humans/data#Eve>	
<http://ns.inria.fr/humans/data#David>	
<http://ns.inria.fr/humans/data#John>	14
<http://ns.inria.fr/humans/data#Karl>	7
<http://ns.inria.fr/humans/data#Mark>	8
<http://ns.inria.fr/humans/data#William>	10
<http://ns.inria.fr/humans/data#Laura>	

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#>
select ?person ?shoesize ?shirtsize where { ?person a h:Person ; h:shoesize ?shoesize ; h:shirtsize ?shirtsize .
filter( (  ?shoesize>8 )  ||  ( ?shirtsize >12  ) ) }
```

Result:



?person	?shoesize	?shirtsize
<http://ns.inria.fr/humans/data#John>	14	12
<http://ns.inria.fr/humans/data#William>	10	13

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 : <http://ns.inria.fr/humans/schema#>
select ?parent where {?parent :hasChild ?child .}

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

One, Gaston who has 2 child Pierre and Jack

2. Find a way to avoid duplicates.

Query:

prefix : <http://ns.inria.fr/humans/schema#>
select distinct ?parent where {?parent :hasChild ?child .}

How many answers do you get then?

4

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

Query:

prefix : <http://ns.inria.fr/humans/schema#>
select ?x ?kind where {?x a ?kind . filter (not exists {?x :hasChild ?child })} values (?kind) { (:Person) (:Man) (:Woman) }

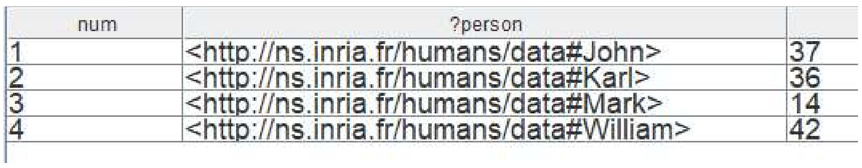
Question 7

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 : <http://ns.inria.fr/humans/schema#>
select ?person ?age where { ?person a :Person; :age ?age}

Result:



num	?person	
1	<http://ns.inria.fr/humans/data#John>	37
2	<http://ns.inria.fr/humans/data#Karl>	36
3	<http://ns.inria.fr/humans/data#Mark>	14
4	<http://ns.inria.fr/humans/data#William>	42

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

Query:

prefix : <http://ns.inria.fr/humans/schema#>
select ?person ?age where { ?person a :Person; :age ?age . filter (?age <18)}

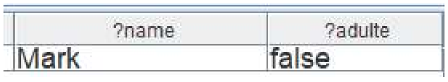
How many answers do you get?

One

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 : <http://ns.inria.fr/humans/schema#>
prefix h: <http://ns.inria.fr/humans/data#>
select  ?name ?adulte where  { h:Mark :age ?age; :name ?name  . bind(IF(?age>18, true,false) as ?adulte )}
```



?name	?adulte
Mark	false

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

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
select  ?name ?age ?even_age where  {
?person a :Person;:age ?age; :name ?name  .
bind( IF(ROUND(?age/2)=?age/2,true,false)  as ?even_age )
}
```

Question 8

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

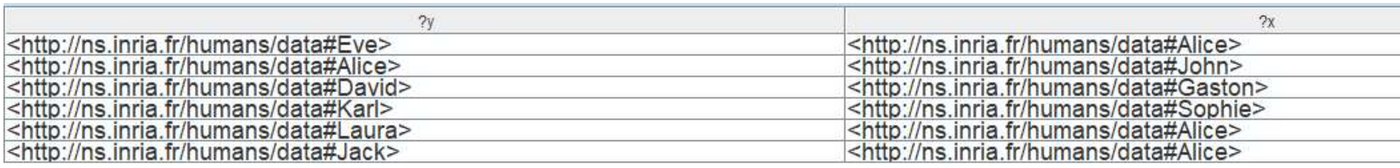
Query:

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

CONSTRUCT {?x :hasFriend ?y}

WHERE {?y :hasFriend ?x}
```

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

Query:

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

SELECT  * WHERE {?y :hasFriend ?x}
```



?y	?x
<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#Alice>
<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/data#Alice>
<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Alice>

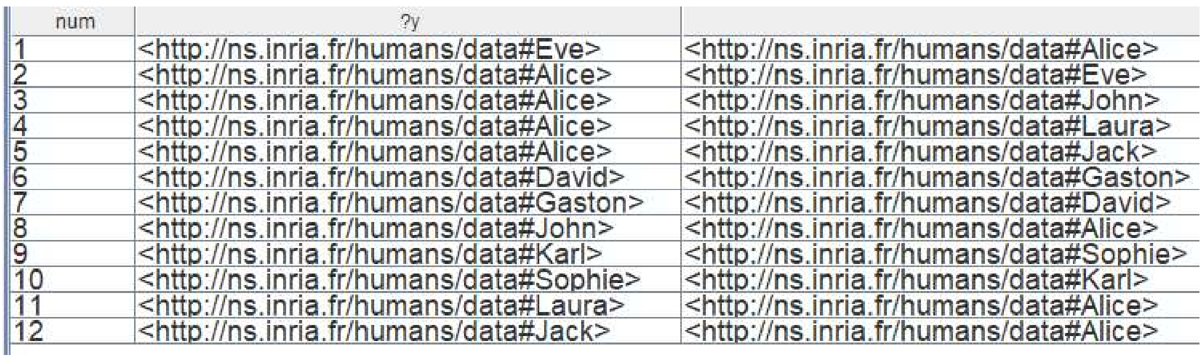
Inserting the value

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

INSERT   {?x :hasFriend ?y}  WHERE {?y :hasFriend ?x}
```

Checking the result:

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

select * WHERE {?y :hasFriend ?x}



num	?y	
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#Alice>
2	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#Eve>
3	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#Laura>
5	<http://ns.inria.fr/humans/data#Alice>	<http://ns.inria.fr/humans/data#Jack>
6	<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/data#Gaston>
7	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/data#David>
8	<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Alice>
9	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Sophie>
10	<http://ns.inria.fr/humans/data#Sophie>	<http://ns.inria.fr/humans/data#Karl>
11	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/data#Alice>
12	<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Alice>

Question 9

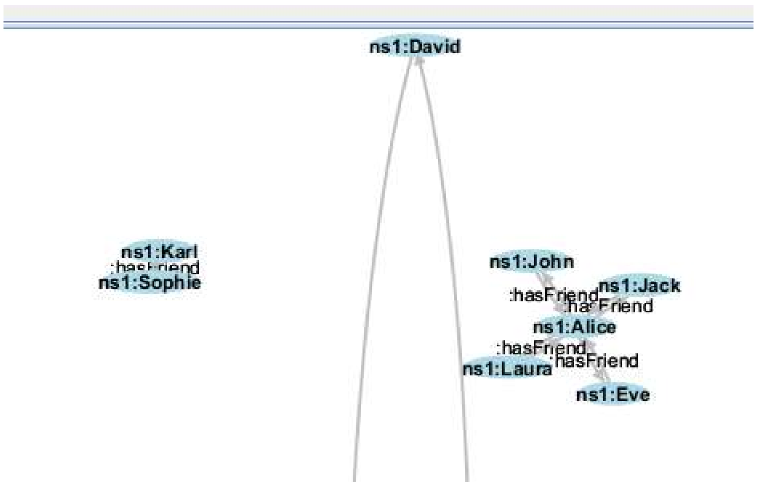
Choose and edit one of the SELECT WHERE queries previously written to transform them into a CONSTRUCT WHERE query (retaining the same WHERE clause) in order to visualize the results as a graph.

Query:

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

Construct {?y :hasFriend ?x} where {SELECT * WHERE {?y :hasFriend ?x}}

Result:

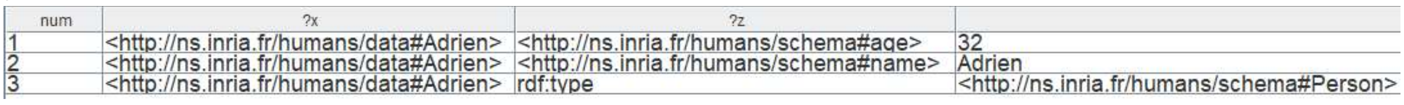


Question 10

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:

```
prefix : <http://ns.inria.fr/humans/schema#>
prefix d: <http://ns.inria.fr/humans/data#>
INSERT data { d:Adrien :name "Adrien"; a :Person; :age 32 .};
select * where {
 ?x ?z ?y
filter (?x== <http://ns.inria.fr/humans/data#Adrien>)
}
```



num	?x	?z	
1	<http://ns.inria.fr/humans/data#Adrien>	<http://ns.inria.fr/humans/schema#age>	32
2	<http://ns.inria.fr/humans/data#Adrien>	<http://ns.inria.fr/humans/schema#name>	Adrien
3	<http://ns.inria.fr/humans/data#Adrien>	rdf:type	<http://ns.inria.fr/humans/schema#Person>

Question 11

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

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
select * where {
  ?x :shirtsize?y .
?z :shirtsize ?y.
filter (?x!=?z)
}
```

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

Query:

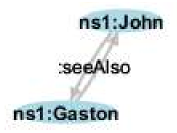
```
prefix : <http://ns.inria.fr/humans/schema#>
construct {?x :seeAlso ?z}
where {
  ?x :shirtsize?y .
?z :shirtsize ?y.
filter (?x!=?z)
}
```

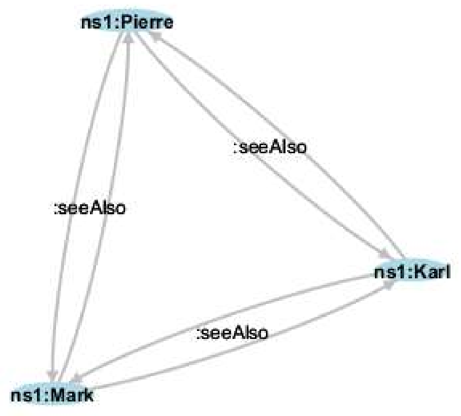
3. Change the query into an insert.

```
prefix : <http://ns.inria.fr/humans/schema#>
insert {?x :seeAlso ?z}
where {
  ?x :shirtsize?y .
?z :shirtsize ?y.
filter (?x!=?z)
}
```

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

screenshot:





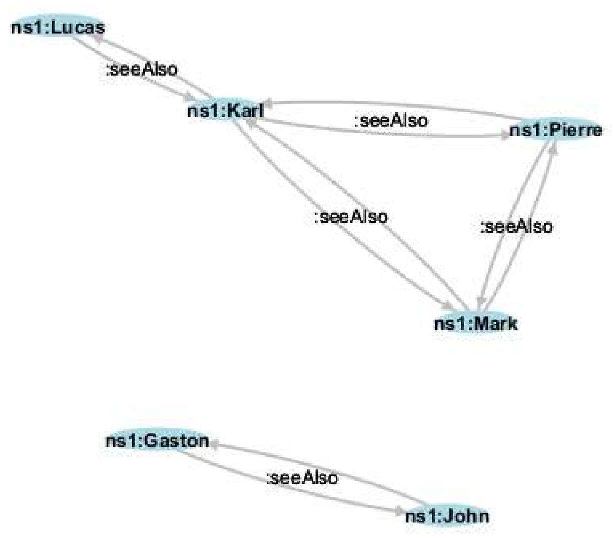
5. Adapt the first query to find persons who have the same shoe size and insert a seeAlso relationship between them.

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
insert {?x :seeAlso ?z}
where {
 ?x :shoesize?y .
?z :shoesize ?y.
filter (?x!=?z)
}
```

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:

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

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

Question 12

1. Find the largest shoe size

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
select (max(?y) as ?m) where { ?x :shoesize ?y}
```

We get 14

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

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
select ?x
where {
{select (max(?y) as ?m) where { ?x :shoesize ?y}}
?x :shoesize ?m .
}
```

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

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
select (avg(?y) as ?m) where { ?x :shoesize ?y}
```

9.28571

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

Query:

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

select (sum(?y)/count(?y) as ?m) where { ?x :shoesize ?y}
```

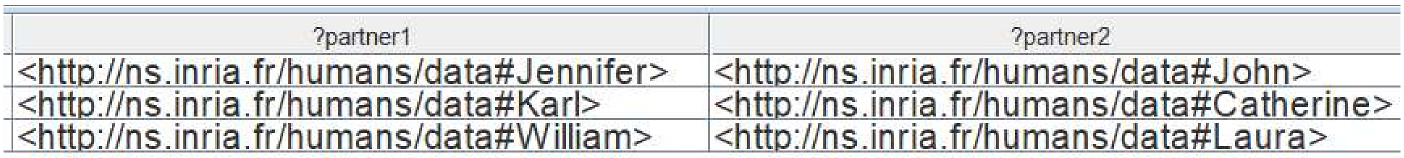
We get the same result

Question 13

Find couples without children

Query:

```
prefix : <http://ns.inria.fr/humans/schema#>
select * where {
    ?partner1 :hasSpouse ?partner2 .
    filter ( not exists {?partner1 :hasChild ?x } && not exists {?partner2 :hasChild ?y}
            && not exists { ?z (:hasParent| :hasMother | :hasFather) ?partner1}
            && not exists { ?v (:hasParent| :hasMother | :hasFather) ?partner1}
    )
}
```



?partner1	?partner2
<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/data#Laura>

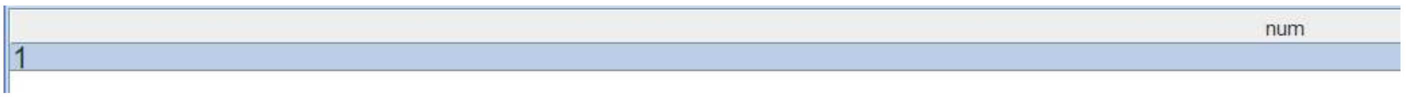
Question 14

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

Insert:

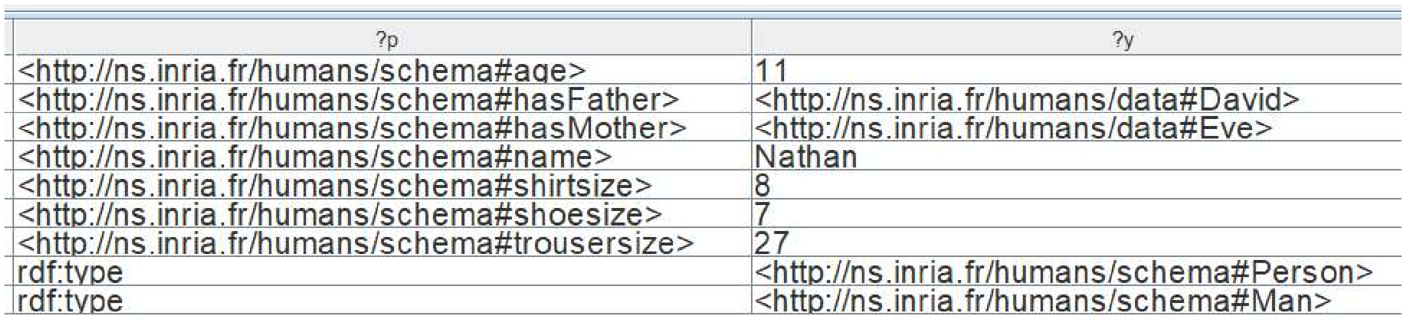
```
prefix : <http://ns.inria.fr/humans/schema#>
prefix h:<http://ns.inria.fr/humans/data#>
INSERT DATA {
    h:Nathan :hasMother h:Eve;
            :hasFather h:David;
            :name "Nathan";
            :shirtsize 8;
            :shoesize 7;
            :age 11;
            :trousersize 27;
            a :Man;
            a :Person .
}
```

Screenshot result:



num
1

```
prefix h:<http://ns.inria.fr/humans/data#>
select * where {  {h:Nathan ?p ?y .} }
```



?p	?y
<http://ns.inria.fr/humans/schema#age>	11
<http://ns.inria.fr/humans/schema#hasFather>	<http://ns.inria.fr/humans/data#David>
<http://ns.inria.fr/humans/schema#hasMother>	<http://ns.inria.fr/humans/data#Eve>
<http://ns.inria.fr/humans/schema#name>	Nathan
<http://ns.inria.fr/humans/schema#shirtsize>	8
<http://ns.inria.fr/humans/schema#shoesize>	7
<http://ns.inria.fr/humans/schema#trousersize>	27
rdf:type	<http://ns.inria.fr/humans/schema#Person>
rdf:type	<http://ns.inria.fr/humans/schema#Man>

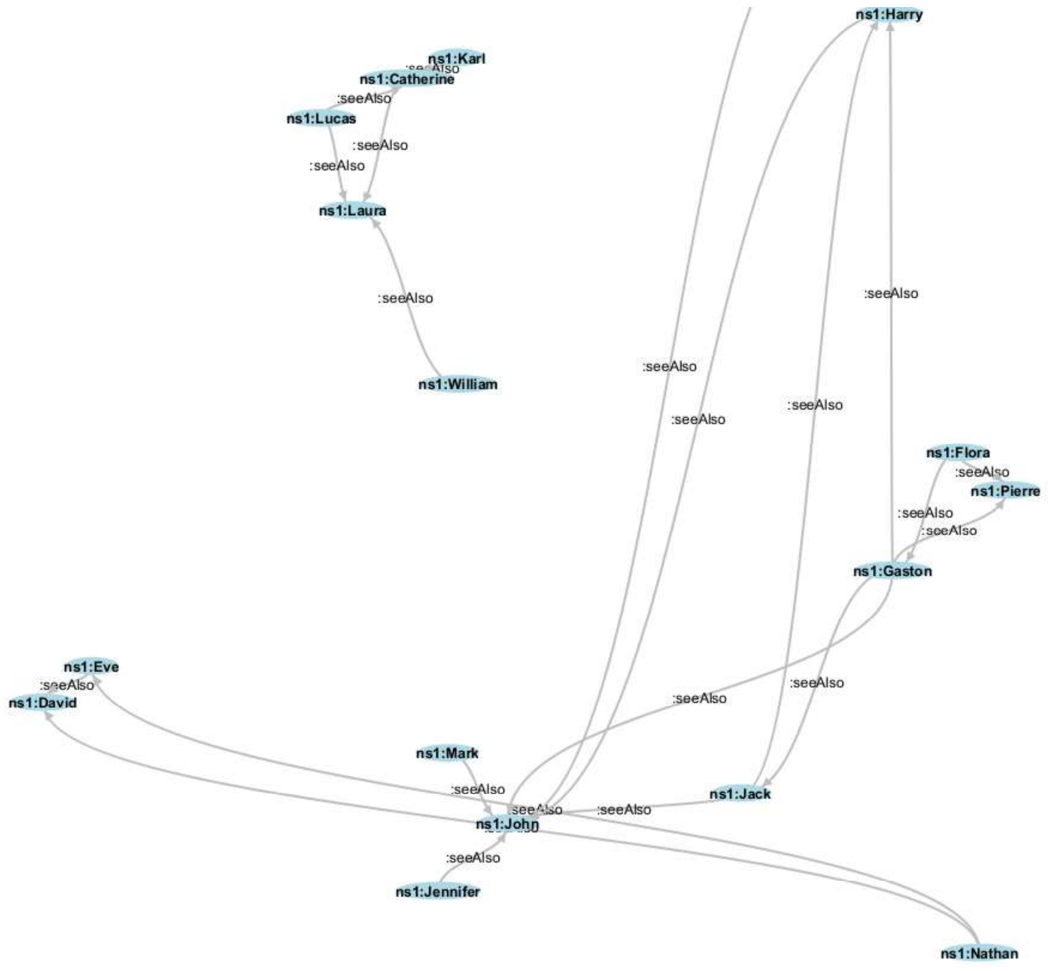
Question 15

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

query:

```
prefix : <http://ns.inria.fr/humans/schema#>
construct { ?x :seeAlso ?y }
where {?x (:hasChild+ | :hasFather+ | :hasMother+ | :hasParent+ |:hasSpouse) ?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

It construct a new graph that include the names of couple of the 20 first couple of the dataset retrieved at the URI http://fr.dbpedia.org/sparql with the service key word.

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

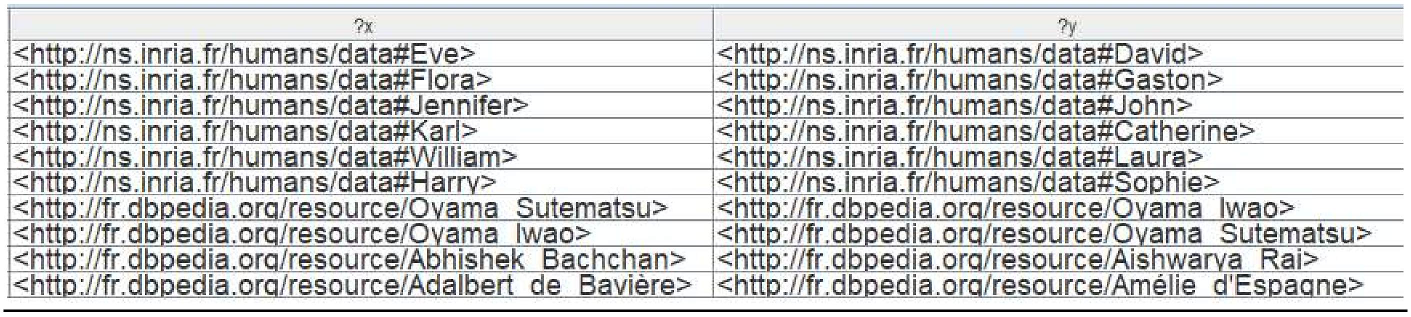
query:

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
  }
}
```

They are new data in the base :

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



?x	?y
<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#David>
<http://ns.inria.fr/humans/data#Flora>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>
<http://fr.dbpedia.org/resource/Oyama Sutematsu>	<http://fr.dbpedia.org/resource/Oyama Iwao>
<http://fr.dbpedia.org/resource/Oyama Iwao>	<http://fr.dbpedia.org/resource/Oyama Sutematsu>
<http://fr.dbpedia.org/resource/Abhishek Bachchan>	<http://fr.dbpedia.org/resource/Aishwarya Rai>
<http://fr.dbpedia.org/resource/Adalbert de Bavière>	<http://fr.dbpedia.org/resource/Amélie d'Espagne>

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 default namespace is http://ns.inria.fr/humans/schema#. It's specidied by the line : a owl:Ontology ;

: refers to the default namespace: @prefix :   <http://ns.inria.fr/humans/schema#> .

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?

Property has the namespace rdf (<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)

Label,comment, Class, subClassOf, domain, range, subpropertyOf have the namespace rdfs (http://www.w3.org/2000/01/rdf-schema#)

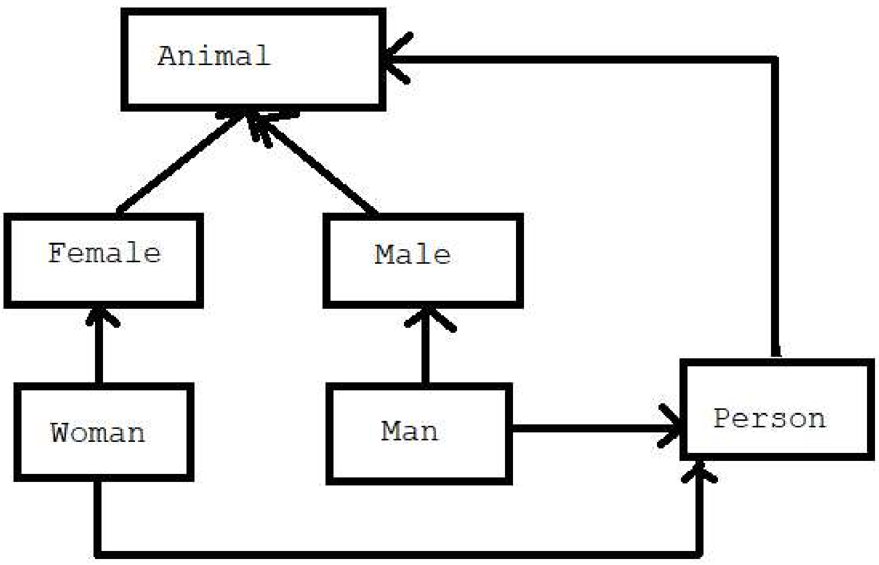
Ontology and priorVersion have the namespace owl (http://www.w3.org/2000/01/rdf-schema#)

4. What are the classes of resources that can have the age property? Explain

All the class can have the age property as no domain or range are given for this property

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:



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

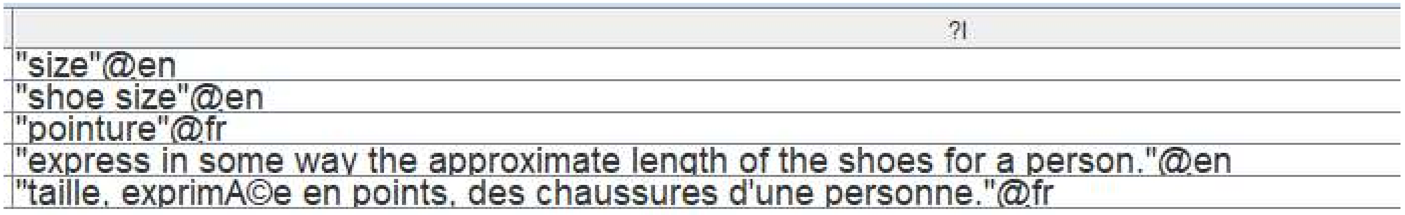
2. Write a query to find all the links subClassOf in the ontology.

query:
select * where {?subClass rdfs:subClassOf ?class}

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 : <http://ns.inria.fr/humans/schema#> .
select * where {
:shoesize rdfs:label|rdfs:comment ?l .
}

answers:



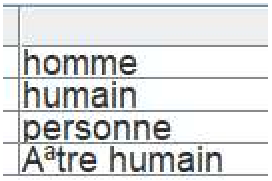
?l
"size"@en
"shoe size"@en
"pointure"@fr
"express in some way the approximate length of the shoes for a person."@en
"taille, exprimA©e en points, des chaussures d'une personne."@fr

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 :     <http://ns.inria.fr/humans/schema#> .
select ?stripped_l where {
   {select ?l where {
   :Person rdfs:label  ?l .
   filter (langMatches(lang(?l),"fr"))}
   }
bind(str(?l) as ?stripped_l)
}
```

answers:



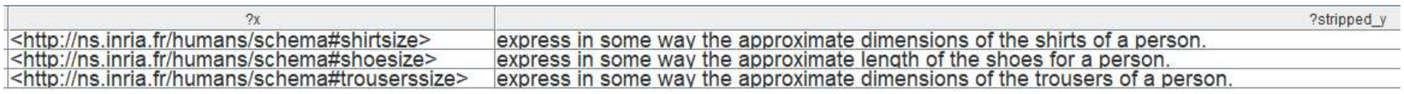
homme
humain
personne
Aªtre humain

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 :     <http://ns.inria.fr/humans/schema#> .
select ?x ?stripped_y where {
?x rdfs:label "size"@en ;rdfs:comment ?y .
filter langMatches(lang(?y),"en")
bind (str(?y) as ?stripped_y)
}
```

answers:

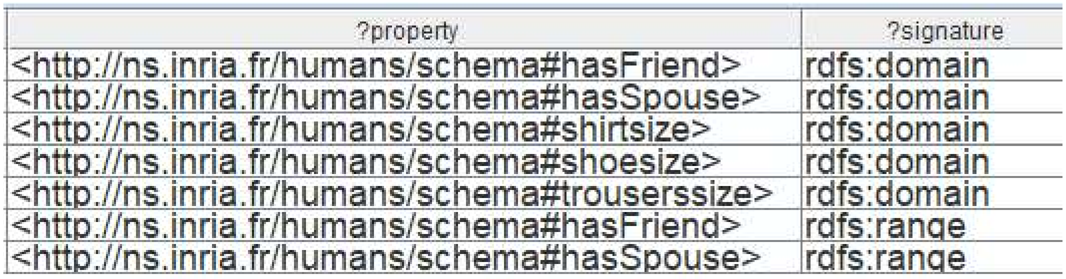


?x	?stripped_y
<http://ns.inria.fr/humans/schema#shirtsize>	express in some way the approximate dimensions of the shirts of a person.
<http://ns.inria.fr/humans/schema#shoesize>	express in some way the approximate length of the shoes for a person.
<http://ns.inria.fr/humans/schema#trouserssize>	express in some way the approximate dimensions of the trousers of a person.

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

query:

```
@prefix : <http://ns.inria.fr/humans/schema#> .
select * where {
        ?x ?r :Person .
        filter ( ?r in (rdfs:domain, rdfs:range))
}
```



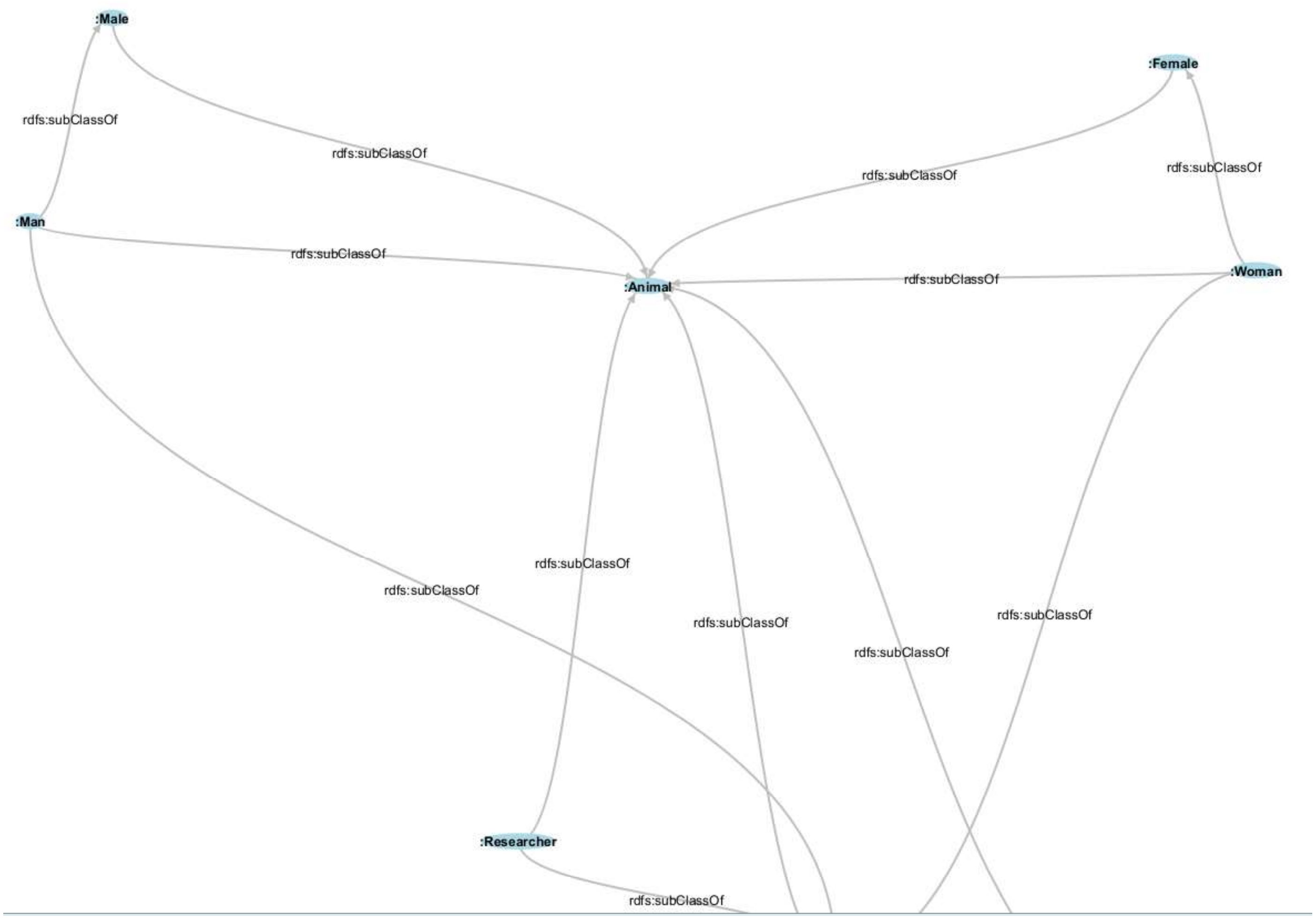
?property	?signature
<http://ns.inria.fr/humans/schema#hasFriend>	rdfs:domain
<http://ns.inria.fr/humans/schema#hasSpouse>	rdfs:domain
<http://ns.inria.fr/humans/schema#shirtsize>	rdfs:domain
<http://ns.inria.fr/humans/schema#shoesize>	rdfs:domain
<http://ns.inria.fr/humans/schema#trouserssize>	rdfs:domain
<http://ns.inria.fr/humans/schema#hasFriend>	rdfs:range
<http://ns.inria.fr/humans/schema#hasSpouse>	rdfs:range

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

query:

```
@prefix : <http://ns.inria.fr/humans/schema#> .
construct {?inheritFrom rdfs:subClassOf ?class} where {
      ?inheritFrom (rdfs:subClassOf) ?class
      FILTER(STRSTARTS(STR(?inheritFrom), STR(:)))
}
```

screenshot:

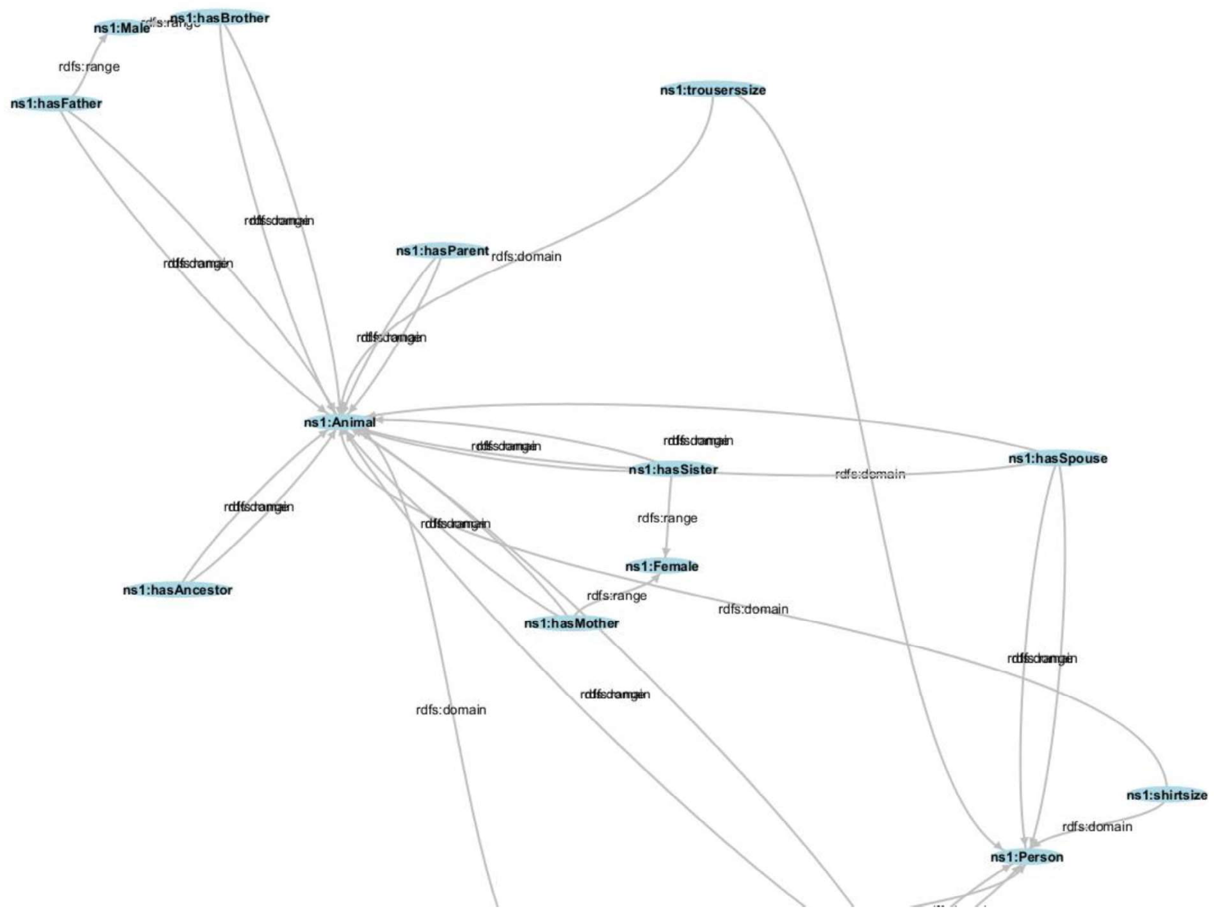


7. To the previous CONSTRUCT add the signatures of the relations.

query:

```
construct {?inheritFrom rdfs:subClassOf ?class. ?z rdfs:domain ?dom. ?z rdfs:range ?range}
where {
        {?inheritFrom (rdfs:subClassOf) ?class FILTER(STRSTARTS(STR(?inheritFrom), STR(:)))}
        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:
@prefix : <http://ns.inria.fr/humans/schema#> .
select * where { ?x a :Person .}

Number of results before:
7

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:
17

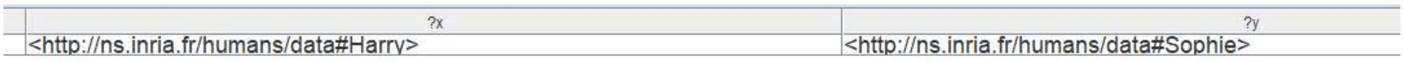
In humans_schema.ttl Lecturer, Man, Woman and Researcher are subclass of Person so the query retrieves also the URI with this type.

Question 2

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

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

Number of results and explanation:
1



?x	?y
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

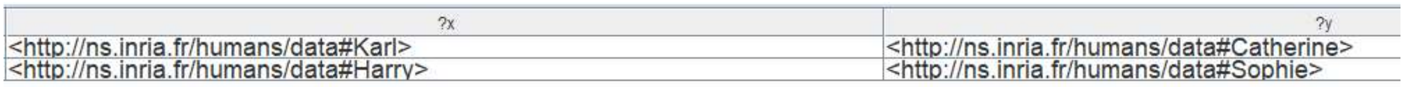
Harry is the only to have the type Man and the relation hasSpouse. As Man is a subclass of Male in humans_schema.ttl, the query get it.

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: one line, after 2 line.
Karl | Catherine is added to the result:



?x	?y
<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

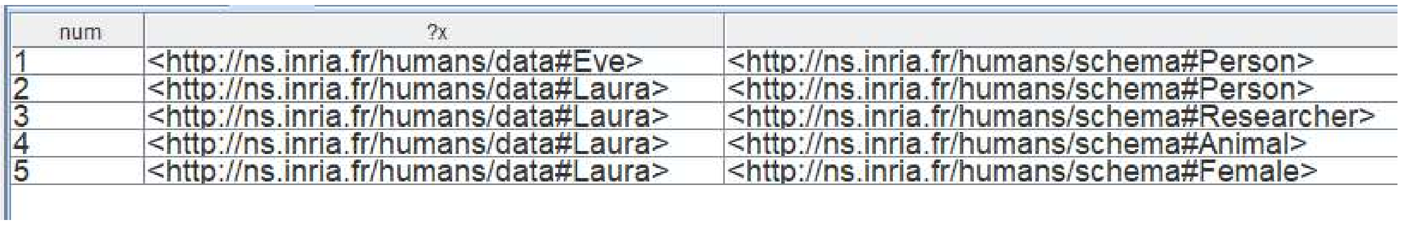
The range of hasFather is :Male so Karl is now declared as Male so the query get him.

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 : <http://ns.inria.fr/humans/schema#> .
select * where { ?x a :Lecturer . ?x a ?y . filter(?y != :Lecturer)}
```

Number of results and your explanation:
5 results (7 without omitting the type Lecturer)



num	?x	
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/schema#Person>
2	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Person>
3	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Researcher>
4	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Animal>
5	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/schema#Female>

Eve is defined with the Lecturer and Person types and Laura with Lecturer, Person and Researcher so the data can't explain the Animal and Female types. Laura is the target of the relation hasMother from Catherine so she as the type Female as this type is define as the range of hasMother. This explain the Female type of Laura. hasMother is also a subProperty of the class hasParent that is a subProperty of hasAncestor and has Ancestor has :Animal for signature so Laura is define as Animal which exlplain the type 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 : <http://ns.inria.fr/humans/schema#> .
select * where { ?x a :Person, :Male .}
```

Your explanation of the result:

jack is of Type Man, but Man is a sub Class of Male and also Person, so Jack is :Male and :Person so it is included in the result set

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

Your explanation of the result:

The hasAncestor relation doesn't exist in the data so the schema must explain the result. hasFather and hasMother are subproperties of hasParent which is a subproperty of hasAncestor, so all hasFather and hasMother relation in the data are also hasAncestor relations.

Question 5

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

Query:

```
@prefix : <http://ns.inria.fr/humans/schema#> .
select distinct ?x ?y ?z where {
        { select ?x ?y ?z where {{?x (:hasSpouse) ?y .}
        optional
        { {?z :hasParent ?x .} union {?x :hasChild ?z .} }}}
        union
        { select ?x ?y ?z where {{?y (^:hasSpouse) ?x .}
        optional
        { {?z :hasParent ?y .} union {?y :hasChild ?z .} }}}
}order by asc(?x)
```
(We could add a filter to remove couple without children)

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

Query:
```
@prefix : <http://ns.inria.fr/humans/schema#> .
construct {?x :coreFamily ?z} where {
        { select ?x ?y ?z where {{?x (:hasSpouse) ?y .}
        optional
        { {?z :hasParent ?x .} union {?x :hasChild ?z .} }}}
        union
        { select ?x ?y ?z where {{?y (^:hasSpouse) ?x .}
        optional
        { {?z :hasParent ?y .} union {?y :hasChild ?z .} }}}

}
```

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

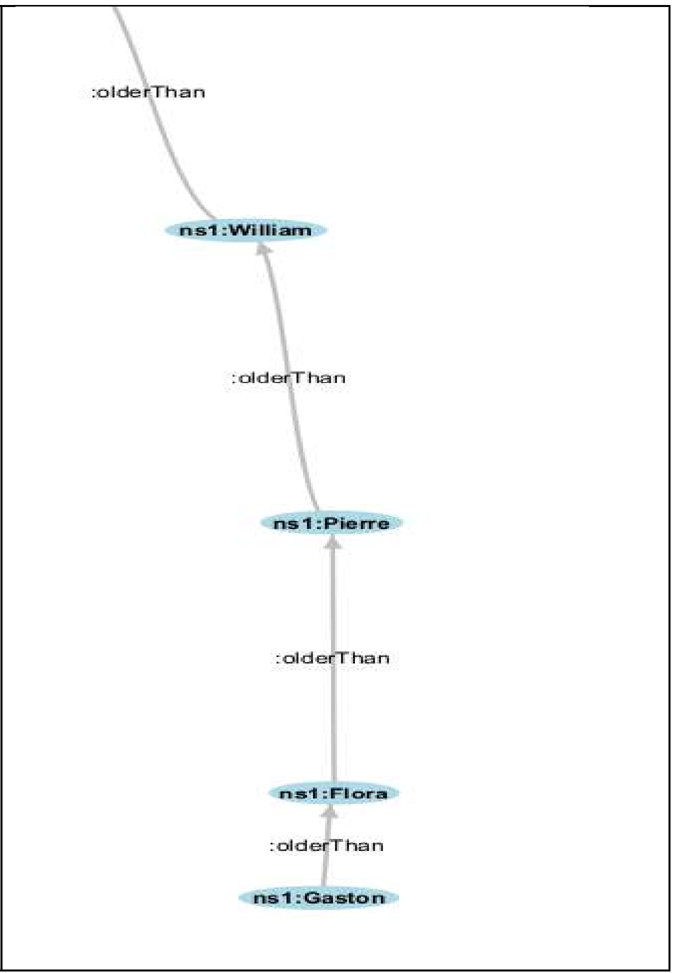
Addition to schema:
```
:olderThan a rdf:Property ;
  rdfs:label "older than" ;
  rdfs:comment "Indicates that one is older than another." ;
  rdfs:domain :Animal ;
  rdfs:range :Animal .
```

Query:
```
construct{
        ?x h:olderThan ?y
}
Where {
        ?x h:age ?a1 .
        ?y h:age ?a2 .
        Filter(?a1 > ?a2)
}
```

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

Query:

```
@prefix : <http://ns.inria.fr/humans/schema#> .
construct {?p2 :olderThan ?p1 } where {
 {select ?p1 ?a1 (min(?a2) as ?ma) where {
   ?p1 :age ?a1 .
   ?p2 :age ?a2 .
 filter (?a2>?a1)} group by ?p1}
?p2 :age ?ma .
}
```



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 h:<http://ns.inria.fr/humans/data#> .
@prefix : <http://ns.inria.fr/humans/schema#> .
select ?prop ?label ?val
where {
            h:John ?prop ?val.
            ?prop rdfs:label ?label .
filter ( contains(?label, "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 h:<http://ns.inria.fr/humans/data#> .
```

```
@prefix : <http://ns.inria.fr/humans/schema#> .
select ?prop ?obj ?p ?val where {
        h:Laura ?prop ?obj.
        ?prop ?p ?val .
        filter (lang(?val)="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.

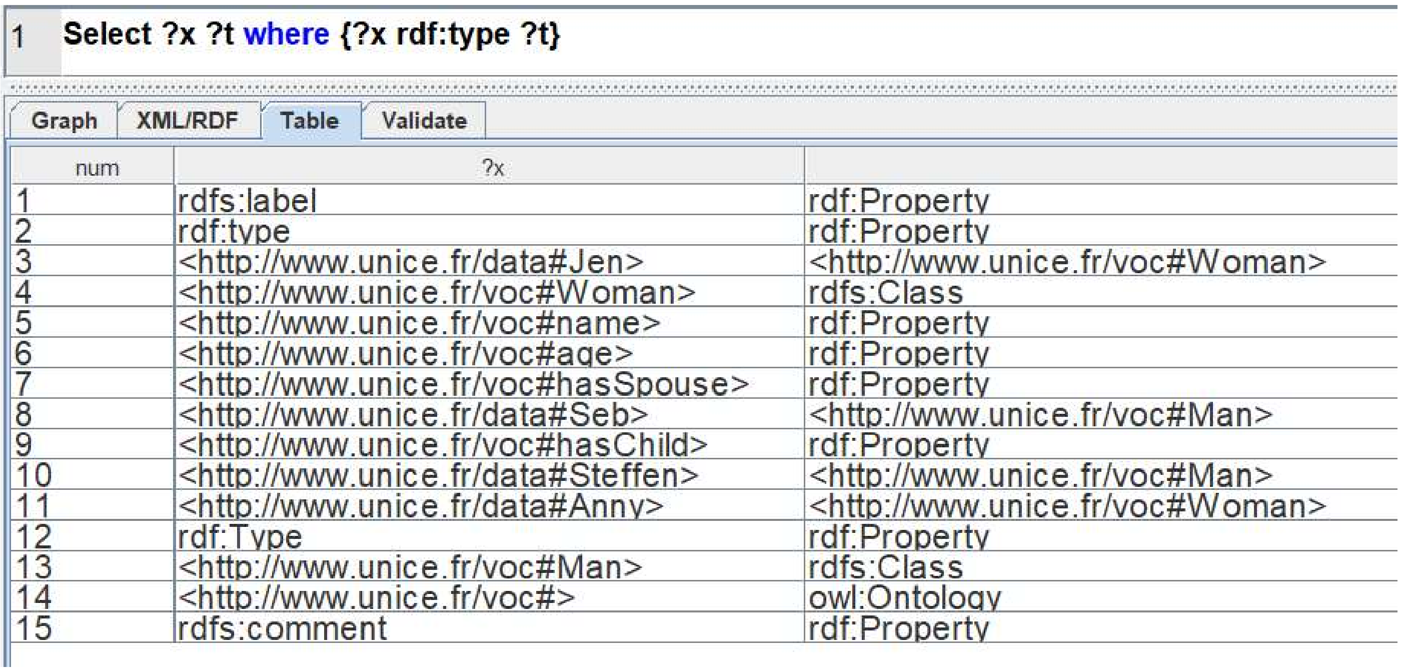
Your schema:

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

: a owl:Ontology ;
  rdfs:label "Family Ontology" ;
  rdfs:comment "An example of schema for RDFS lab" .

:Woman a rdfs:Class ;
  rdfs:label "woman"@en, "femme"@fr .

:Man a rdfs:Class ;
  rdfs:label "man"@en, "homme"@fr .

:name a rdf:Property ;
  rdfs:label "name"@en, "nom"@fr ;
  rdfs:comment "designation of something."@en,
        "désignation de quelque chose."@fr .

:age a rdf:Property ;
  rdfs:label "age"@en, "âge"@fr ;
  rdfs:comment "complete existence duration."@en,
        "durée complète d'existence."@fr .

:hasChild a rdf:Property ;
  rdfs:label "has for child"@en, "a pour enfant"@fr .

:hasSpouse a rdf:Property ;
  rdfs:label "has for spouse"@en, "est en ménage avec"@fr ;
  rdfs:comment "a person's partner in marriage."@en,
        "le partenaire d'une personne dans un mariage."@fr .
```

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:



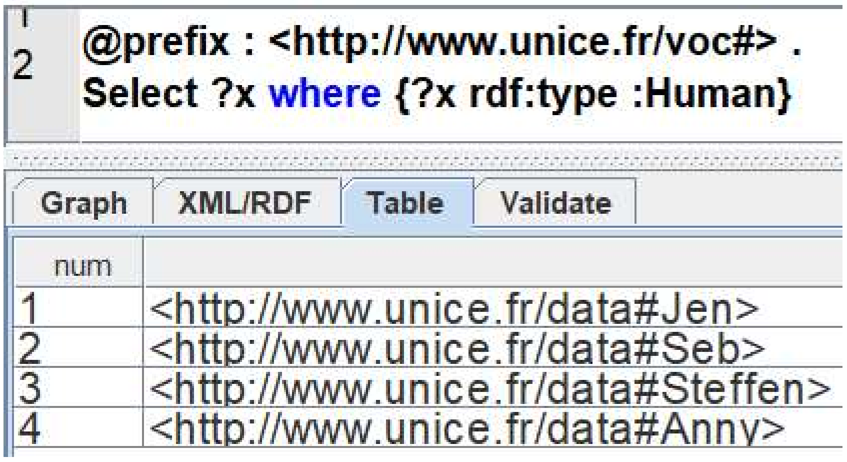
```
Select ?x ?t where {?x rdf:type ?t}
```

Graph | XML/RDF | Table | Validate

num	?x	
1	rdfs:label	rdf:Property
2	rdf:type	rdf:Property
3	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/voc#Woman>
4	<http://www.unice.fr/voc#Woman>	rdfs:Class
5	<http://www.unice.fr/voc#name>	rdf:Property
6	<http://www.unice.fr/voc#age>	rdf:Property
7	<http://www.unice.fr/voc#hasSpouse>	rdf:Property
8	<http://www.unice.fr/data#Seb>	<http://www.unice.fr/voc#Man>
9	<http://www.unice.fr/voc#hasChild>	rdf:Property
10	<http://www.unice.fr/data#Steffen>	<http://www.unice.fr/voc#Man>
11	<http://www.unice.fr/data#Anny>	<http://www.unice.fr/voc#Woman>
12	rdf:Type	rdf:Property
13	<http://www.unice.fr/voc#Man>	rdfs:Class
14	<http://www.unice.fr/voc#>	owl:Ontology
15	rdfs:comment	rdf:Property

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:



```
@prefix : <http://www.unice.fr/voc#> .
Select ?x where {?x rdf:type :Human}
```

Graph | XML/RDF | Table | Validate

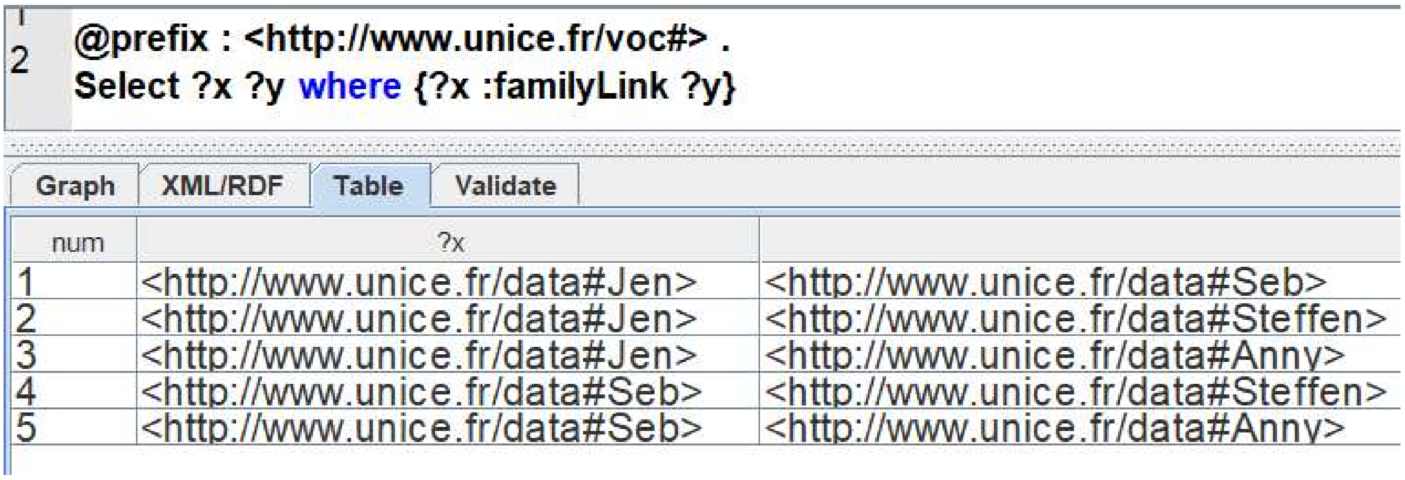
num	
1	<http://www.unice.fr/data#Jen>
2	<http://www.unice.fr/data#Seb>
3	<http://www.unice.fr/data#Steffen>
4	<http://www.unice.fr/data#Anny>

Explanation:

Man and Woman are subclass of Human so per type propagation, all Man and Woman are also 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:



```
@prefix : <http://www.unice.fr/voc#> .
Select ?x ?y where {?x :familyLink ?y}
```

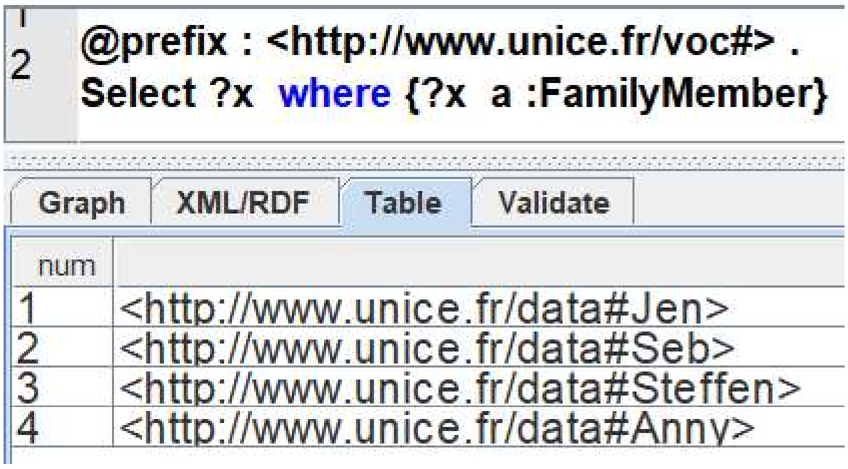
num	?x	
1	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/data#Seb>
2	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/data#Steffen>
3	<http://www.unice.fr/data#Jen>	<http://www.unice.fr/data#Anny>
4	<http://www.unice.fr/data#Seb>	<http://www.unice.fr/data#Steffen>
5	<http://www.unice.fr/data#Seb>	<http://www.unice.fr/data#Anny>

Explanation:

hasChild and hasSpouse are sub property of familyLink so by type propagation they are also familyLink

- 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:



```
@prefix : <http://www.unice.fr/voc#> .
Select ?x  where {?x  a :FamilyMember}
```

num	
1	<http://www.unice.fr/data#Jen>
2	<http://www.unice.fr/data#Seb>
3	<http://www.unice.fr/data#Steffen>
4	<http://www.unice.fr/data#Anny>

Explanation:

By type inference, all IRI that are at start of a property are of the type of the domain property.
By type inference, all IRI that are at the end of a property are of the type of the range property.
As the familyLink has MemberFamily for domain and range and hasSpouse, hasChild are sub property of familyLink they also have the same range and domain. So by type inference of domain and range, all IRI linked by one of those properties are also MemberFamily

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.

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` .

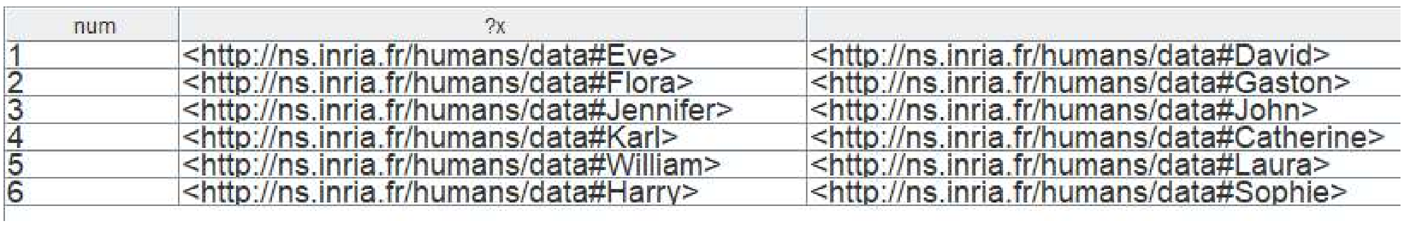
Code added to the schema:

```
:hasSpouse a rdf:Property ;
   a owl:SymmetricProperty ;
   rdfs:label "has for spouse"@en, "est en mÃ©nage avec"@fr ;
   rdfs:comment "a person's partner in marriage."@en,
            "le partenaire d'une personne dans un mariage."@fr ;
   rdfs:domain :Person ;
   rdfs:range  :Person .
```

Query:

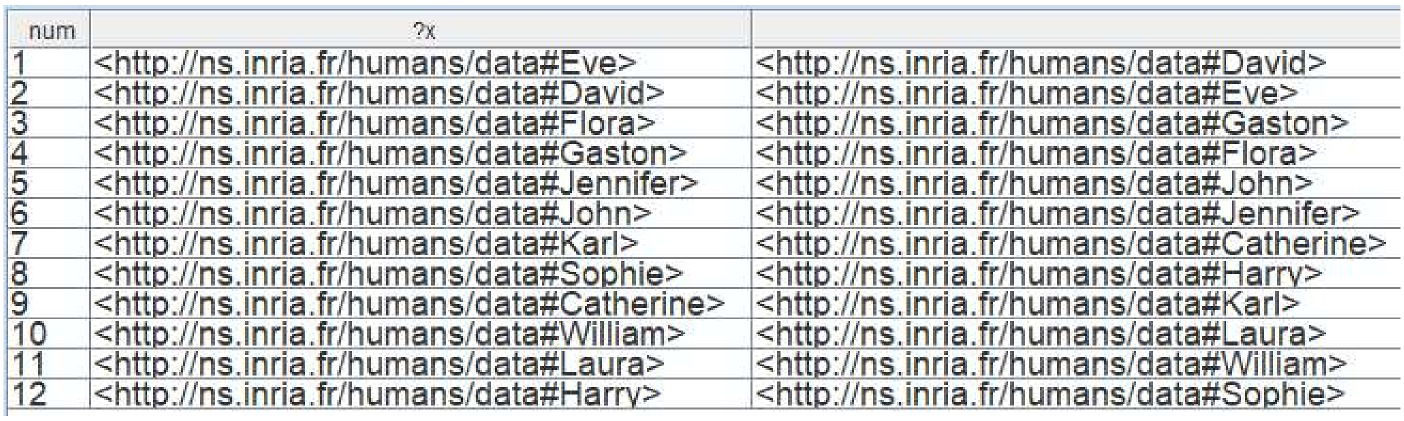
```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?x :hasSpouse  ?y .}
```

Result before addition to the schema:



num	?x	
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#David>
2	<http://ns.inria.fr/humans/data#Flora>	<http://ns.inria.fr/humans/data#Gaston>
3	<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/data#John>
4	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
5	<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/data#Laura>
6	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

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



num	?x	
1	<http://ns.inria.fr/humans/data#Eve>	<http://ns.inria.fr/humans/data#David>
2	<http://ns.inria.fr/humans/data#David>	<http://ns.inria.fr/humans/data#Eve>
3	<http://ns.inria.fr/humans/data#Flora>	<http://ns.inria.fr/humans/data#Gaston>
4	<http://ns.inria.fr/humans/data#Gaston>	<http://ns.inria.fr/humans/data#Flora>
5	<http://ns.inria.fr/humans/data#Jennifer>	<http://ns.inria.fr/humans/data#John>
6	<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Jennifer>
7	<http://ns.inria.fr/humans/data#Karl>	<http://ns.inria.fr/humans/data#Catherine>
8	<http://ns.inria.fr/humans/data#Sophie>	<http://ns.inria.fr/humans/data#Harry>
9	<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Karl>
10	<http://ns.inria.fr/humans/data#William>	<http://ns.inria.fr/humans/data#Laura>
11	<http://ns.inria.fr/humans/data#Laura>	<http://ns.inria.fr/humans/data#William>
12	<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Sophie>

Explanation:

As the relation is now symmetrical i.e works in both direction, the number of result has doubled

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

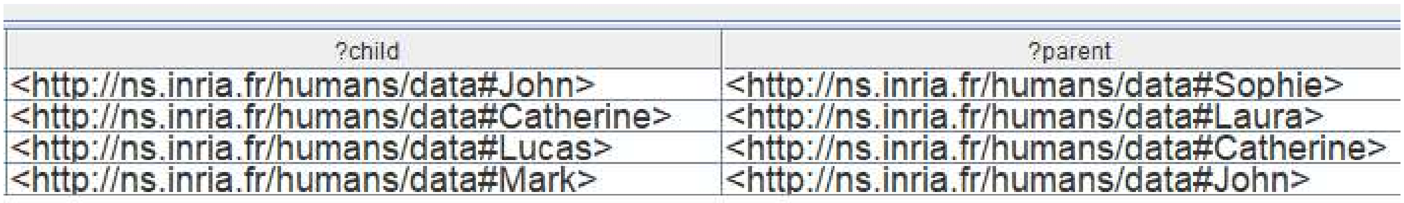
Code added to the schema:

```
:hasChild a rdf:Property ;
  rdfs:label "has for child"@en, "a pour enfant"@fr ;
  rdfs:comment "relation between an animal and another animal to which it gave birth."@en,
          "relation entre un animal et un autre animal auquel il a donnÃ© naissance."@fr ;
 owl:inverseOf :hasParent .
```

Query:

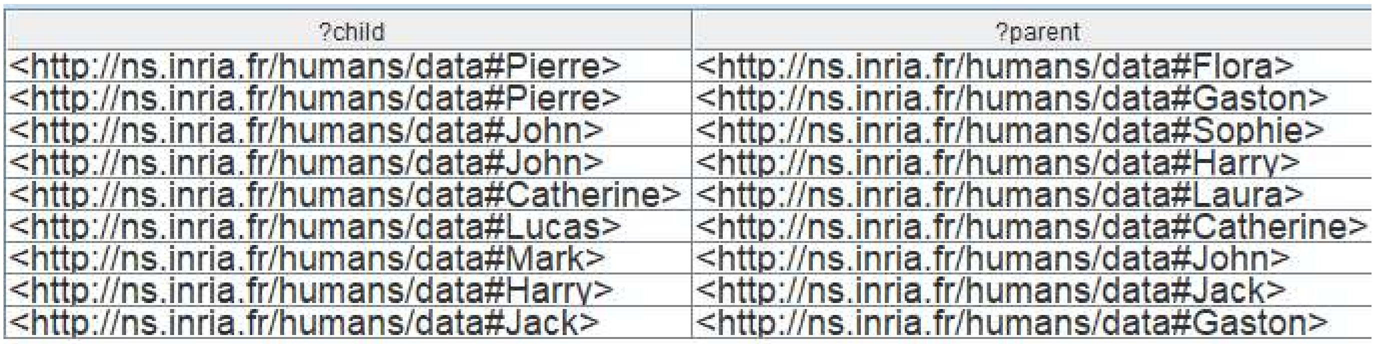
```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?child :hasParent  ?parent .}
```

Result before addition to the schema:



?child	?parent
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#John>

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



?child	?parent
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Flora>
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Harry>
<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Jack>
<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Gaston>

Explanation:

hasChild now means targets of the relationship are also the start of hasParent relationship so we now have as many hasParent relationship as hasChild + the declared hasParent relationship

3. Declare `hasAncestor` as transitive property.

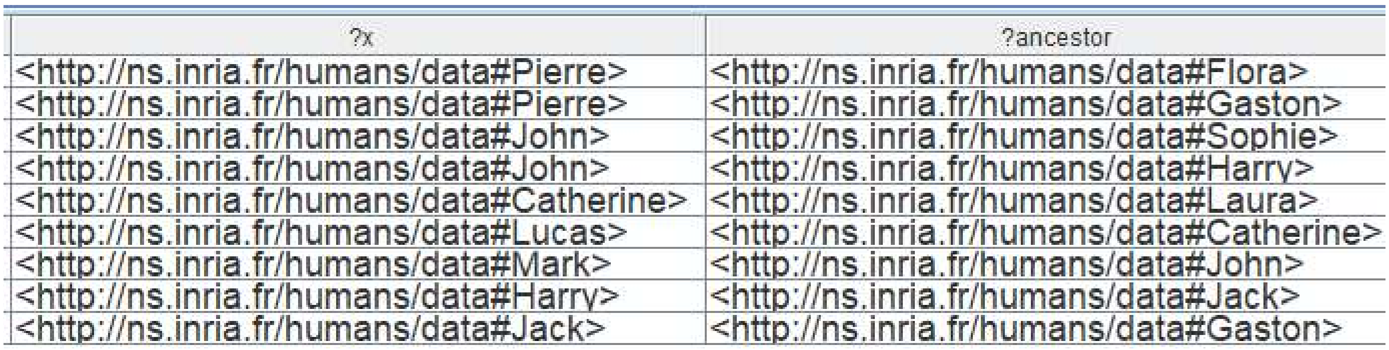
Code added to the schema:

```
:hasAncestor a rdf:Property ;
   rdfs:label "has for ancestor"@en, "a pour ancÃªtre"@fr ;
   rdfs:comment "relation between an animal and another animal from which it is descended."@en,
            "relation entre un animal et un autre animal duquel il descend."@fr ;
   rdfs:domain :Animal ;
   rdfs:range  :Animal ;
   a owl:TransitiveProperty .
```

Query:

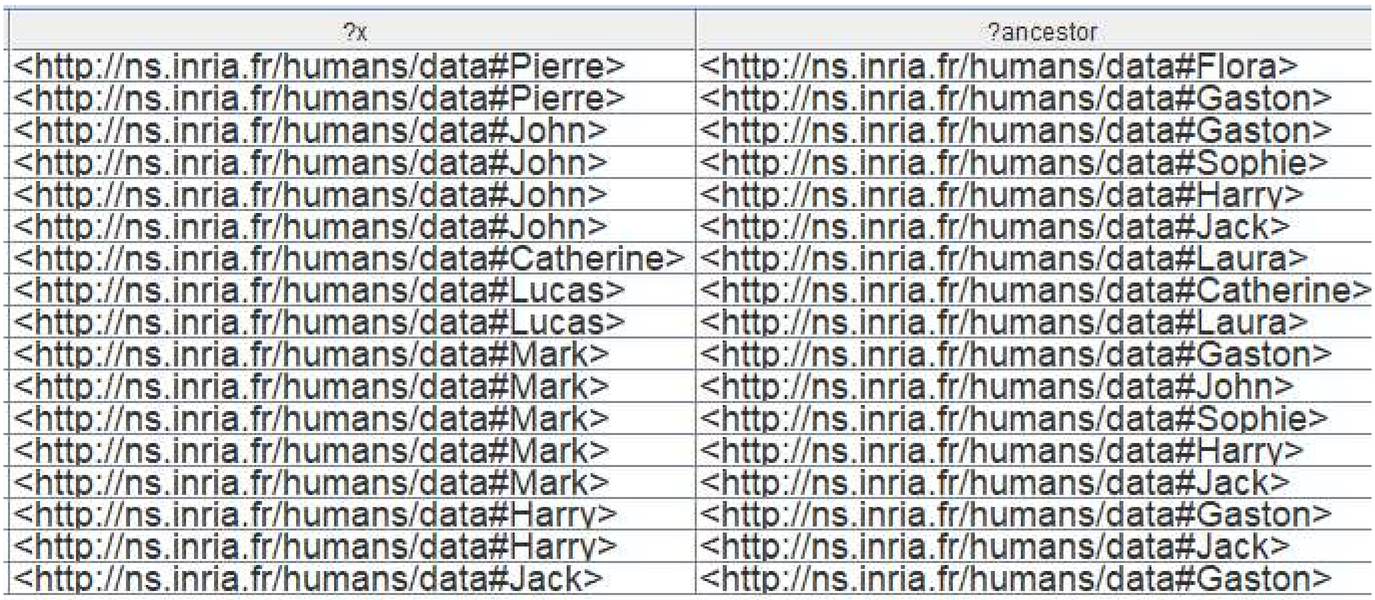
```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?x :hasAncestor  ?ancestor .}
```

Result before addition to the schema:



?x	?ancestor
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Flora>
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Harry>
<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Jack>
<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Gaston>

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



?x	?ancestor
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Flora>
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Harry>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Jack>
<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Harry>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#Jack>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Jack>
<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Gaston>

Explanation:

All consecutive path along the hasAncestor are considered: if A has a relationship hasAncestor with B and B as the same relationship with C then A has B and C as ancestor.

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

Code added to the schema:

```
:hasSibling a owl:ObjectProperty ;
   owl:propertyChainAxiom ( :hasParent :hasChild ).
```

Query:

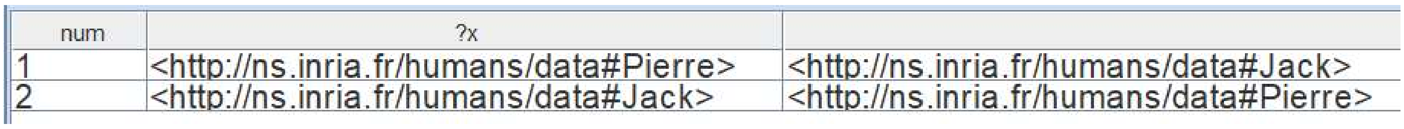
```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?x (:hasSister|:hasBrother)  ?y .}
```

Result before addition to the schema:

No result

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

```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?x (:hasSibling)  ?y . filter (?x!=?y)}
```



num	?x	
1	<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Jack>
2	<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Pierre>

Explanation:

The has sibling property is chained so if a node x is linked to a node y by a hasParent property and y is linked by a hasChild property to z then x is linked to z by the property hasSibling so even if no hasSister or hasBrother is define in the data we find hasSibling relation.

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:

```
:hasUncle rdf:type owl:ObjectProperty ;
    owl:propertyChainAxiom ( :hasParent :hasBrother ) .

:hasAunt rdf:type owl:ObjectProperty ;
    owl:propertyChainAxiom ( :hasParent :hasSister ) .
```

Query:

```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?x ?p  ?y filter( ?p in ( :hasAunt, :hasUncle ) )}
```

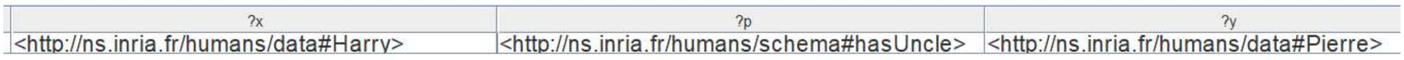
Result before addition to the schema (reload then unselect and reselect “OWL RL”):

None

(although the query

```
prefix :<http://ns.inria.fr/humans/schema#>
select distinct ?x ?z where {
    ?x  (:hasParent|^:hasChild)  ?y .
    ?y (:hasBrother|:hasSister) ?z .
} find the same result that below minus the property type)
```

Result after addition to the schema:



?x	?p	?y
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/schema#hasUncle>	<http://ns.inria.fr/humans/data#Pierre>

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

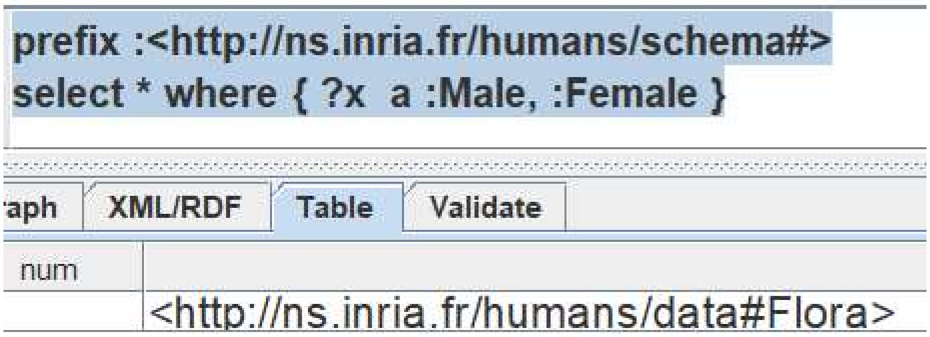
Code added to the schema:

```
:Female a rdfs:Class ;
   rdfs:label "female"@en, "femelle"@fr ;
   rdfs:comment "an animal that produces gametes (ova) that can be fertilized by male gametes
(spermatozoa)."@en,
             "animal appartenant au sexe apte à produire des ovules."@fr ;
   rdfs:subClassOf :Animal;
     owl:disjointWith :Male .

: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#>
select * where { ?x  a :Male, :Female }
```

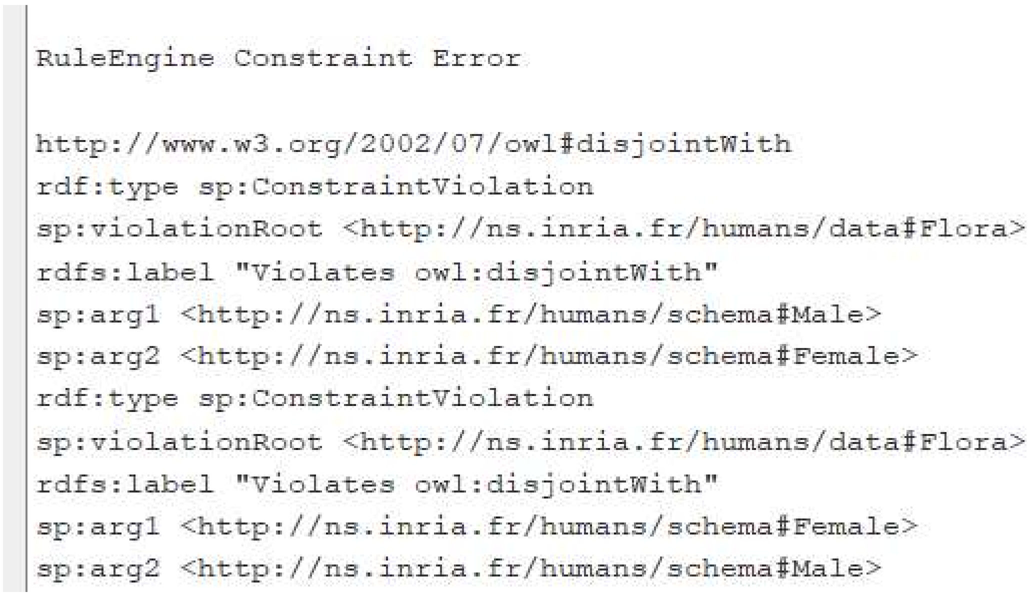
Result before addition to the schema:



```
prefix :<http://ns.inria.fr/humans/schema#>
select * where { ?x  a :Male, :Female }
```

```
num
     <http://ns.inria.fr/humans/data#Flora>
```

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



```
RuleEngine Constraint Error

http://www.w3.org/2002/07/owl#disjointWith
rdf:type sp:ConstraintViolation
sp:violationRoot <http://ns.inria.fr/humans/data#Flora>
rdfs:label "Violates owl:disjointWith"
sp:arg1 <http://ns.inria.fr/humans/schema#Male>
sp:arg2 <http://ns.inria.fr/humans/schema#Female>
rdf:type sp:ConstraintViolation
sp:violationRoot <http://ns.inria.fr/humans/data#Flora>
rdfs:label "Violates owl:disjointWith"
sp:arg1 <http://ns.inria.fr/humans/schema#Female>
sp:arg2 <http://ns.inria.fr/humans/schema#Male>
```

Explanation:

With the disjoint property between :Male and Female no node can be of this two type at the same time so the validation throws an error when the definition of Flora is read as she is declared of both types.

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

Code added to the schema:

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

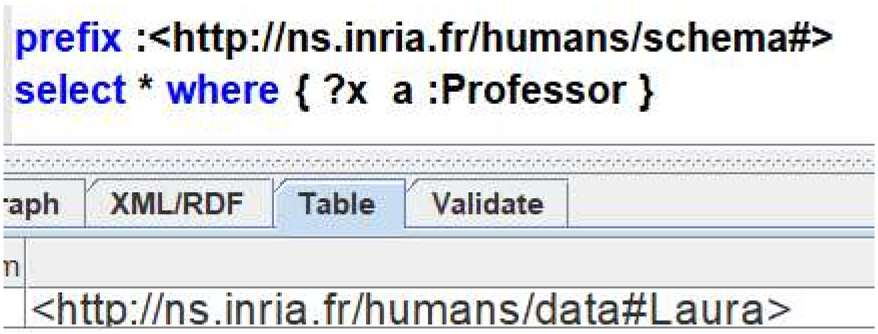
Query:

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

Result before addition to the schema:

none

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



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

```
n
<http://ns.inria.fr/humans/data#Laura>
```

Explanation:

Laura is both a Lecturer and a Researcher so by definition of professor she is also a profesor

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

Code added to the schema:

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

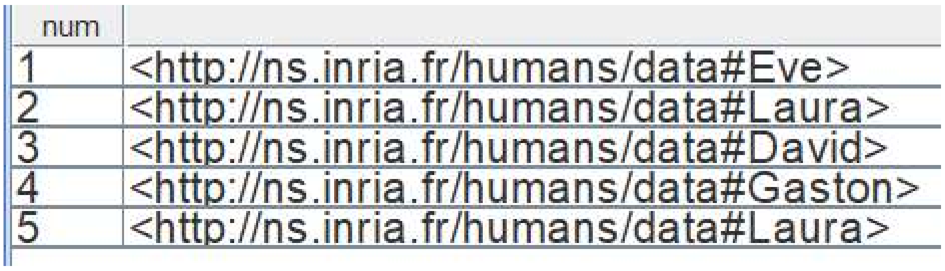
Query:

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

Result before addition to the schema:

None but the query

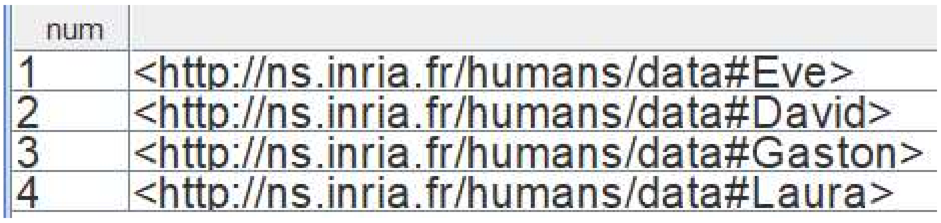
prefix :<http://ns.inria.fr/humans/schema#>
select ?x where {   {?x a :Lecturer .} union {?x a :Researcher .} } returns



num	
1	<http://ns.inria.fr/humans/data#Eve>
2	<http://ns.inria.fr/humans/data#Laura>
3	<http://ns.inria.fr/humans/data#David>
4	<http://ns.inria.fr/humans/data#Gaston>
5	<http://ns.inria.fr/humans/data#Laura>

We see that Laura appears two times

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



num	
1	<http://ns.inria.fr/humans/data#Eve>
2	<http://ns.inria.fr/humans/data#David>
3	<http://ns.inria.fr/humans/data#Gaston>
4	<http://ns.inria.fr/humans/data#Laura>

Explanation:

All those nodes are Researcher or Lecturer so they are also Academic by definition. (Laura appears only one time without the need of distinct with this method)

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):

```
:Professor a rdfs:Class; owl:intersectionOf (:Lecturer :Researcher);
    rdfs:subClassOf [
                 a owl:Restriction;
                        owl:onProperty :mainEmployer;
                        owl:allValuesFrom :University] .

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

:Organization a rdfs:Class .

:University a rdfs:Class;
    rdfs:subClassOf :University .

:mainEmployer a rdfs:Property;
    rdfs:domain :Person;
    rdfs:range :Organization .
```

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 "inria".
```

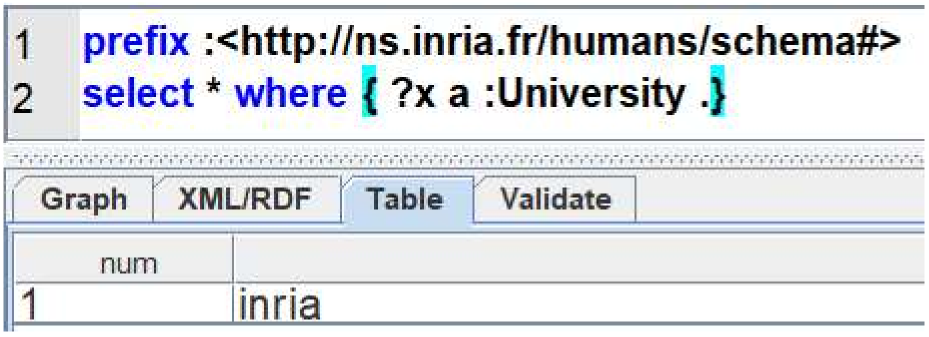
Query:

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

Result before addition to the schema:

none

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



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

num	
1	inria

Explanation:

The restriction on Professor class implies that all values of the property mainEmployer for this class are University therefore “inria” is of University class by inference.

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:

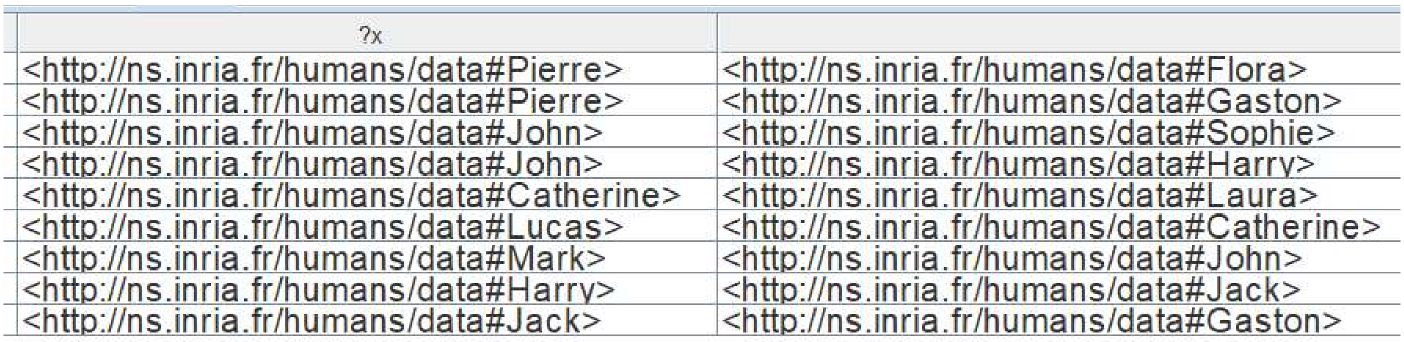
```
:Person a rdfs:Class ;
  rdfs:label "human"@en, "human being"@en, "person"@en,
           "homme"@fr, "humain"@fr, "personne"@fr, "être humain"@fr ;
  rdfs:comment "a member of the human species"@en,
            "un membre de l'espèce humaine."@fr ;
  rdfs:subClassOf :Animal;
    owl:equivalentClass [ a owl:Restriction;
                                  owl:onProperty :hasParent ;
                                  owl:someValuesFrom :Woman ] .
```

Query:

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

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

Result before addition to the schema:



?x	
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Flora>
<http://ns.inria.fr/humans/data#Pierre>	<http://ns.inria.fr/humans/data#Gaston>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Sophie>
<http://ns.inria.fr/humans/data#John>	<http://ns.inria.fr/humans/data#Harry>
<http://ns.inria.fr/humans/data#Catherine>	<http://ns.inria.fr/humans/data#Laura>
<http://ns.inria.fr/humans/data#Lucas>	<http://ns.inria.fr/humans/data#Catherine>
<http://ns.inria.fr/humans/data#Mark>	<http://ns.inria.fr/humans/data#John>
<http://ns.inria.fr/humans/data#Harry>	<http://ns.inria.fr/humans/data#Jack>
<http://ns.inria.fr/humans/data#Jack>	<http://ns.inria.fr/humans/data#Gaston>

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

The same

Explanation:

I don't understand why the schema doesn't generate an error. It should because some Person doesn't have female parent e.g Harry has parent Jack but Jack is a man and man is disjoint with female so Jack can't be a woman. Per the w3.org documentation of owl, the someValuesFrom exclusion implies in our case that for all Persons, they have at least one parent that is a woman but we don't observe this as some person doesn't have woman parent and we shouldn't be able to infer that they have. No man are of type Woman after adding the restriction. Also corese doesn't find any rule to apply

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 — An os must be Windows, Mac or Linux

   :Os a owl:Class ; owl:oneOf (:Windows :Mac :Linux) .

2. owl:unionOf — Genre is the union of Male and female

   :Genre a owl:Class; owl:unionOf (:Male :Female) .

3. owl:intersectionOf — An hermaphrodite is both male and female

   :Hermaphrodite a owl:Class; owl:intersectionOf (:Male :Female) .

4. owl:complementOf — Dry is the complement of moist
   :Dry owl:Class; owl:complementOf :Moist .

5. owl:disjointWith — Unbroken is disjoint of broken
   or owl:AllDisjointClasses
   or owl:disjointUnionOf

   :Unbroken a owl:Class; owl:disjointWith :Broken .

6. owl:ObjectProperty — ownCar links a human and its car

   :ownCar a owl:ObjectProperty .

7. owl:DatatypeProperty — has income gives the income of a person

   :hasIncome a owl:DatatypeProperty .

8. owl:SymmetricProperty — inFinalWith is the links between two finalist person
   or owl:AsymmetricProperty

   inFinalWith a owl:SymmetricProperty .

9. owl:inverseOf — looseTo links a loser to its winner

   :looseTo owl:inverseOf :winAgainst .

10. owl:TransitiveProperty — harderThan means one material is harder than an other

    :harderThan owl:TransitiveProperty .

11. owl:propertyDisjointWith hasTested is disjoint with wantToTest

```
:hasTested owl:propertyDisjointWith :wantToTest .
```

12. owl:ReflexiveProperty sameBirthdayAs: peoples birthed the same day
 or owl:IrreflexiveProperty

```
:sameBirthdayAs a owl:ReflexiveProperty .
```

13. owl:propertyChainAxiom worksInBuilding property means an employee works in a building. It's the chain of an employee working in a department of an enterprise that is in building. Therefore the employee works in the same building.

```
:worksInBuilding owl:propertyChainAxiom ( :worksInDepartment :isLocatedIn ) .
```

14. owl:FunctionalProperty :species is the species of an animal

```
:species a owl:FunctionalProperty .
```

15. owl:InverseFunctionalProperty hasFingerPrint define the finger print of a person

```
:hasFingerPrint a owl:inverseFunctionalProperty .
```

16. owl:hasKey one person is identified by its fingerprint and its DNA

```
:Person owl:hasKey (:hasFingerPrint :hasDna ) .
```

17. owl:allValuesFrom Carnivore eat only meat

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

18. owl:someValuesFrom Cake need at leat one baker

```
:Cake a owl:Class;
owl:subClassOf [a owl:Restriction ;
                owl:onProperty :hasBaker ;
                owl:someValuesFrom :Baker ] .
```

19. owl:hasValue biped has 2 legs

```
:Biped a owl:Class;
owl:equivalentClass [a owl:Restriction ;
                owl:onProperty :nbLegs ;
                owl:hasValue: "2" ] .
```

20. owl:maxCardinality House owner have at least one house
 or owl:minCardinality

```
:HouseOwner a owl:Class;
        Owl:subClassOf [a owl:Restriction ; owl:minCardinality "1"; owl:onProperty :ownHouse ] .
```

21. owl:qualifiedCardinality			Book has a price

```
:Book a owl:Class;
        Owl:subClassOf [a owl:Restriction ; owl:qualifiedCardinality “1”; owl:onProperty :price ] .
```