PRACTICAL SESSIONS

by : **Yassine ASSAFI**
Email : **Yassine.assafi@edu.dsti.institute**

Day 02: Answers to the practical session on RDF.

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 RDF online translator: http://rdf-translator.appspot.com/
- The SPARQL Corese engine: https://project.inria.fr/corese/

Create RDF

Read carefully the following statements:

> "Jen is a 42-year old woman and she has a shoe size of 36 and trouser size of 38. She is, married to Seb who is a man with whom she had two children: Anny who is a woman and Steffen who is a man. Jen is also an engineer and Catherine and Fabien are her colleagues. Jen's father is a man named Thomas"

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
5. Use the RDF online translator to validate your N3 and translate it into RDF/XML: http://rdf-translator.appspot.com/
6. Compare your RDF/XML with the result of the N3 translation
7. Translate in other formats to see the results.

Code of validated RDF in N3 syntax:

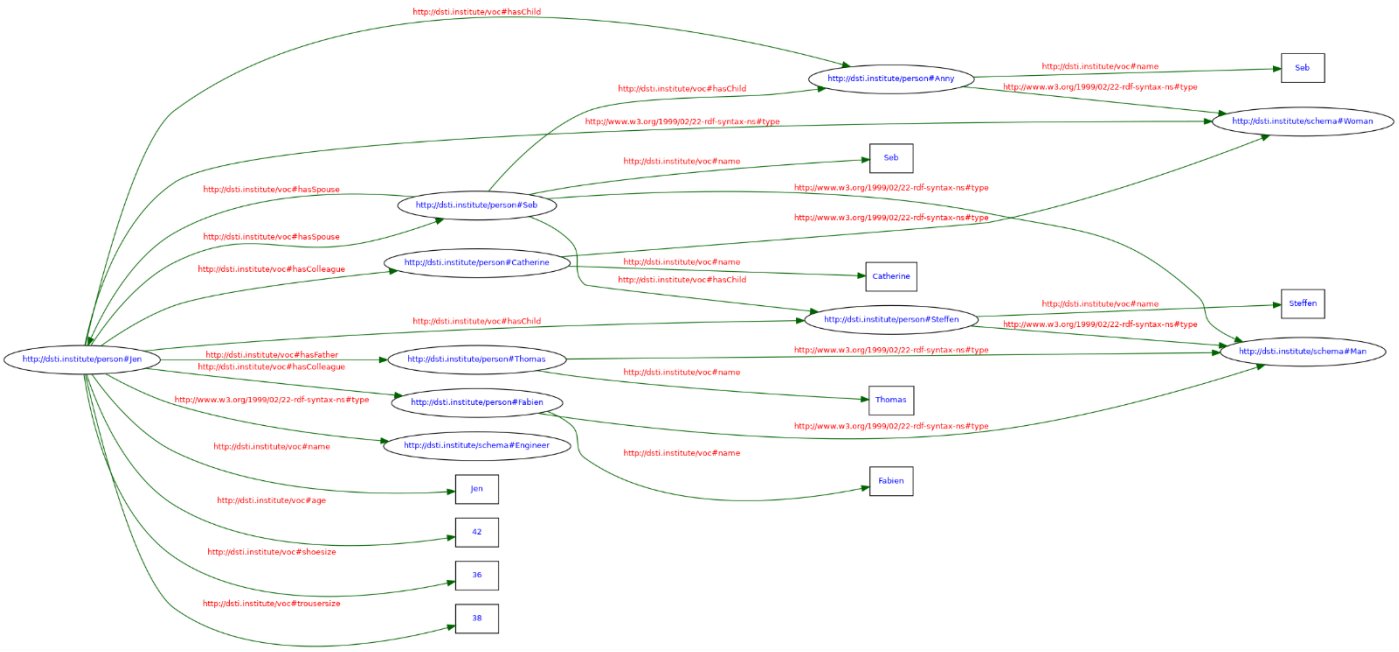
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix : <http://dsti.institute/Family.rdfs#> .
@base <http://dsti.institute/Family.rdfs-instances>.

<#Jen> a :Woman, :Engineer;
        :name "Jen"^^xsd:string;
        :age "42"^^xsd:integer;
        :shoesize "36"^^xsd:integer;
        :trousersize "38"^^xsd:integer;
```

```
        :hasSpouse <#Seb>;
        :hasChild <#Anny>, <#Steffen>;
        :hasFather <#Thomas>;
        :hasColleage <#Catherine>, <#Fabien>.

<#Seb> a :Man;
        :name "Seb"^^xsd:string;
        :hasChild <#Anny>, <#Steffen>;
        :hasSpouse <#Jen>.

<#Anny> a :Woman;
        :name "Anny"^^xsd:string.

<#Steffen> a :Man;
        :name "Steffen"^^xsd:string.

<#Thomas> a :Man;
        :name "Thomas"^^xsd:string.

<#Catherine> a :Woman;
        :name "Catherine"^^xsd:string.

<#Fabien> a :Man;
        :name "Fabien"^^xsd:string.
```

Code of validated RDF in XML syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [ <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"> <!ENTITY class
"http://dsti.institute/Family.rdfs"> ]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:base=" http://dsti.institute/Family.rdfs-instances"
>

<rdf:Description rdf:about="#Jen">
        <rdf:type rdf:resource="&class;#Woman"/>
        <rdf:type rdf:resource="&class;#Engineer"/>
        <name rdf:datatype="&xsd;string">Jen</name>
        <age rdf:datatype="&xsd;integer">42</age>
        <shoesize rdf:datatype="&xsd;integer">36</shoesize>
        <trousersize rdf:datatype="&xsd;integer">38</trousersize>
        <hasSpouse rdf:resource="#Seb"/>
        <hasChild rdf:resource="#Anny"/>
        <hasChild rdf:resource="#Steffen"/>
        <hasFather rdf:resource="#Thomas"/>
        <hasColleague rdf:resource="#Fabien"/>
        <hasColleague rdf:resource="#Catherine"/>
</rdf:Description>

<rdf:Description rdf:about="#Seb">
        <rdf:type rdf:resource="&class;#Man"/>
        <name rdf:datatype="&xsd;string">Seb</name>
```

```
        <hasSpouse rdf:resource="#Jen"/>
        <hasChild rdf:resource="#Anny"/>
        <hasChild rdf:resource="#Steffen"/>
</rdf:Description>

<rdf:Description rdf:about="#Anny">
        <rdf:type rdf:resource="&class;#Woman"/>
        <name rdf:datatype="&xsd;string">Seb</name>
</rdf:Description>

<rdf:Description rdf:about="#Steffen">
        <rdf:type rdf:resource="&class;#Man"/>
        <name rdf:datatype="&xsd;string">Steffen</name>
</rdf:Description>

<rdf:Description rdf:about="#Thomas">
        <rdf:type rdf:resource="&class;#Man"/>
        <name rdf:datatype="&xsd;string">Thomas</name>
</rdf:Description>

<rdf:Description rdf:about="#Fabien">
        <rdf:type rdf:resource="&class;#Man"/>
        <name rdf:datatype="&xsd;string">Fabien</name>
</rdf:Description>

<rdf:Description rdf:about="#Catherine">
        <rdf:type rdf:resource="&class;#Woman"/>
        <name rdf:datatype="&xsd;string">Catherine</name>
</rdf:Description>

</rdf:RDF>
```



Query your data

Download the Corese.jar library and start it as a standalone application: On Window double-click the file ".jar". If it does not work or on other platforms, 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 two tabs: (1) one to load input files and see traces of execution, and (2) the default tab to start loading or writing queries and see their result. Load the annotations contained in the file "Jen.rdf" you created and validated before. The interface contains a default SPARQL query:

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

The SPARQL language will be presented in the next course. Just know that this query can find all of the resources referred to in the data you loaded and their types. Launch the query and check the results.

Understand existing data

1, Get the RDF/XML about http://ns.inria.fr/fabien.gandon#me and translate the RDF/XML into Turtle/N3

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

Can you guess the link between http://ns.inria.fr/fabien.gandon and http://ns.inria.fr/fabien.gandon#me

The URI http://ns.inria.fr/fabien.gandon refers to a PersonalProfileDocument which is a class that represents the document and provides properties about the person who is the maker and primary topic of the document, that is associated with the URI http://ns.inria.fr/fabien.gandon#me in this case.

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 rdf:type yago:WikicatCapitalsInEurope;

3, If you don't have the human dataset file yet, at the following address you will find an RDF file containing several annotations:

http://wimmics.inria.fr/doc/tutorial/human_2013.rdf

Download the file and use the RDF XML online validation service to validate the XML and see the triples and the graph.

1. What is the namespace used for instances / resources created in this file?
   xml:base="http://www.inria.fr/2007/09/11/humans.rdfs-instances">

2. By which mechanism is the association between instances and namespace done i.e. how was the instance namespace specified?
   By defining the BASE namespace in the root.

3. What is the namespace of the vocabulary used to describe the resources in the dataset and how is it associated with the tags?
   xmlns="http://www.inria.fr/2007/09/11/humans.rdfs#"

4. Explain the code `xmlns="&humans;#"`
   &humans refers to the ENTITY humans : http://www.inria.fr/2007/09/11/humans.rdfs,
   and since xmlns defines a namespace with no 'abbreviation', so all tags (Man, hasChild…) will correspond actually to http://www.inria.fr/2007/09/11/humans.rdfs#tag.

5. Find *everything* about information on John in this file.
   all the information:

   John is a Person
   name : John
   shoesize : 14
   shirtsize : 12
   trousersize : 44
   age : 37
   has parent : Sophie, Harry
   father of : Mark
   friend of : Alice
   spouse of : Jennifer

6. Translate the file in turtle and save it as human_2013.ttl
   10 first lines:

```
@prefix : <http://www.inria.fr/2007/09/11/humans.rdfs#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve> a :Lecturer,
   :Person ;
   :hasFriend <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice> ;
   :hasSpouse <http://www.inria.fr/2007/09/11/humans.rdfs-instances#David> ;
   :name "Eve" .

<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora> a :Woman ;
   :age 95 ;
   :hasChild <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre> ;
   :hasSpouse <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston> ;
   :name "Flora" .
```

7. In the turtle version find *everything* about Laura.
   all the information:
   Laura is a Person, Lecturer and Researcher
   name : Laura
   friend of : Alice
   mother of : Catherine
   spouse of : William

Day 02: Answers to the practical 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: https://project.inria.fr/corese/
- The human dataset file and the human shape file from the archive

What is that shape

With you text editor open the file human_2013_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:

sh:NodeShape

What is the target of that shape:

All instances of the :Person class

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

A Person instance must have at least one name value.

What is the severity level of that constraint?

sh:Violation

In Corese load the dataset human_2013_dataset_rdf.ttl (menu “load RDF”) and this shape (menu “load SHACL”) and run the validation in a query tab (button “SHACL”). Explain in English what the report is saying:

The validation reports indicates false conformity. The :Person Karl is missing a name, so we get a violation.

Add your constraints

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

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

In Corese load the human dataset (menu “load RDF”) and this shape (menu “load SHACL”) and run the validation in a query tab (button “SHACL”). Explain in English shat the report is saying:

The validation reports indicates false conformity. There are 3 warnings and 1 violation. The :Person Karl is missing a name, and the people Eve, David and Laura are missing the age value.

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

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

In Corese load the human dataset (menu “load RDF”) and this shape (menu “load SHACL”) and run the validation in a query tab (button “SHACL”). Explain in English shat the report is saying:

In addition to past 1 violation and 3 warnings, the validation reports indiquates 6 new non-conformities of severity INFO.

Day 03: Answers to the practical 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: https://project.inria.fr/corese/

Basic query on RDF human.rdf

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

On Window double-click the file ".jar". If it does not work or on other platforms, 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 two tabs: (1) one to load input files and see traces of execution, and (2) the default tab to start loading or writing queries and see their result.

If you don't have the human dataset file yet download the following file of annotations and save it as "human.rdf":

http://wimmics.inria.fr/doc/tutorial/human_2013.rdf

Load the file human.rdf as RDF data in corese.

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 the results as RDF triples.

Screenshot:

Question 2:

Create a new tab to enter the following query:

```
prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where { ?x a ?t . filter(strstarts(?t, h:)) }
```

Translate this query in plain English.

Returns all triples for which the resources correspond to Types (ex: Man, Lecturer..) that have the qualified name h:Type.

Run this query. How many answers do you get?

21

Find John and his types in the answers.

John's types: <http://www.inria.fr/2007/09/11/humans.rdfs#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

   PREFIX c: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
   SELECT ?p ?y
   WHERE { { c:John ?p ?y } UNION { ?x ?p c:John } }

   Results:

num	?p	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs#age>	37
2	<http://www.inria.fr/2007/09/11/humans.rdfs#hasParent>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
3	<http://www.inria.fr/2007/09/11/humans.rdfs#name>	John
4	<http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize>	12
5	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	14
6	<http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize>	44
7	rdf:type	<http://www.inria.fr/2007/09/11/humans.rdfs#Person>
8	<http://www.inria.fr/2007/09/11/humans.rdfs#hasChild>	
9	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFather>	
10	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>	
11	<http://www.inria.fr/2007/09/11/humans.rdfs#hasSpouse>	

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

   Query

   DESCRIBE <http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>

   Results:

num	?_ast_p_0	?_ast_v_0	?_ast_v_1	?_ast_p_1
1	<http://www.inria.fr/2007/09/11/hu...	37		
2	<http://www.inria.fr/2007/09/11/hu...	<http://www.inria.fr/2007/09/11/hu...		
3	<http://www.inria.fr/2007/09/11/hu...	John		
4	<http://www.inria.fr/2007/09/11/hu...	12		
5	<http://www.inria.fr/2007/09/11/hu...	14		
6	<http://www.inria.fr/2007/09/11/hu...	44		
7	rdf:type	<http://www.inria.fr/2007/09/11/hu...		
8			<http://www.inria.fr/2007/09/11/hum...	<http://www.inria.fr/2007/09/11/hu...
9			<http://www.inria.fr/2007/09/11/hum...	<http://www.inria.fr/2007/09/11/hu...
10			<http://www.inria.fr/2007/09/11/hum...	<http://www.inria.fr/2007/09/11/hu...
11			<http://www.inria.fr/2007/09/11/hum...	<http://www.inria.fr/2007/09/11/hu...

Question 4

Create a new tab to enter the following query:

```
prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>

select * where { ?x h:hasSpouse ?y }
```

Translate this query in plain English.

Returns all RDF triples that have a hasSpouse relationship.

Run this query. How many answers do you get?

6 triples

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>

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://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?size
WHERE { ?x a h:Person; h:shoesize ?size }
```

Result:

num	?x	
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	8
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	10
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	7

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?size
WHERE { ?x a h:Person. OPTIONAL { ?x h:shoesize ?size} }
```

Result:

num	?x	
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	8
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>	
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	10
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	7

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://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?shoesize?shirtsize
WHERE { ?x a h:Person; h:shoesize ?shoesize; h:shirtsize ?shirtsize. FILTER(?shoesize>=8 || ?shirtsize>=12) }

Result:

num	?x	?shoesize	
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	14	12
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	8	9
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#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 h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE { ?x h:hasChild ?y }

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

5 answers, there is one duplicate.

num	?x
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>

2. Find a way to avoid duplicates.

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT DISTINCT ?x
WHERE { ?x h:hasChild ?y }

How many answers do you get then?

4 answers.

3. Rewrite a query to find the Persons who have no child.

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>

SELECT ?x
WHERE { ?x a h:Person. FILTER (! EXISTS { ?x h:hasChild ?y }) }

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 people with their age.

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?age
WHERE { ?x a h:Person. OPTIONAL {?x h:age ?age} }

Result:

num	?x	
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	37
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	14
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>	
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	42
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	36

2. Formulate a query to find people who are not adults.

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?age
WHERE { ?x a h:Person; h:age ?age. FILTER (?age < 18) }

How many answers do you get?

1 answer.

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://www.inria.fr/2007/09/11/humans.rdfs#>
ASK { ?x a h:Person; h:name "Mark"; h:age ?age. FILTER (?age >= 18) }

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

Query:

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?age ?even
WHERE { ?x a h:Person; h:age ?age BIND(xsd:integer(?age/2)*2=?age AS ?even) }

num	?x	?age	?even
1	<http://www.inria.fr/2007/09/11/...	37	false
2	<http://www.inria.fr/2007/09/11/...	14	true
3	<http://www.inria.fr/2007/09/11/...	42	true
4	<http://www.inria.fr/2007/09/11/...	36	true

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://www.inria.fr/2007/09/11/humans.rdfs#>
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://www.inria.fr/2007/09/11/humans.rdfs#>
INSERT {  ?y h:hasFriend ?x }
WHERE {  ?x h:hasFriend ?y }
```

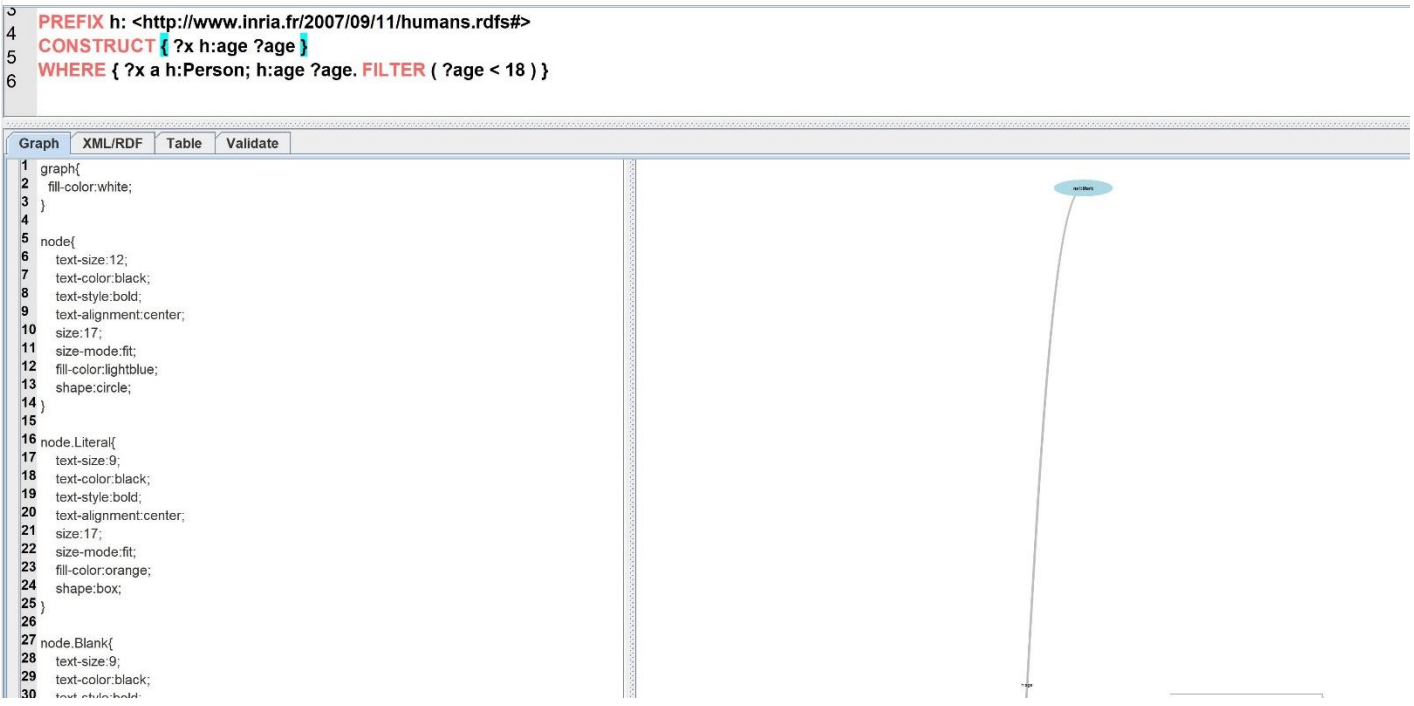
Question 9

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://www.inria.fr/2007/09/11/humans.rdfs#>
CONSTRUCT { ?x h:age ?age }
WHERE { ?x a h:Person; h:age ?age. FILTER ( ?age < 18 ) }
```

Result:



Question 10

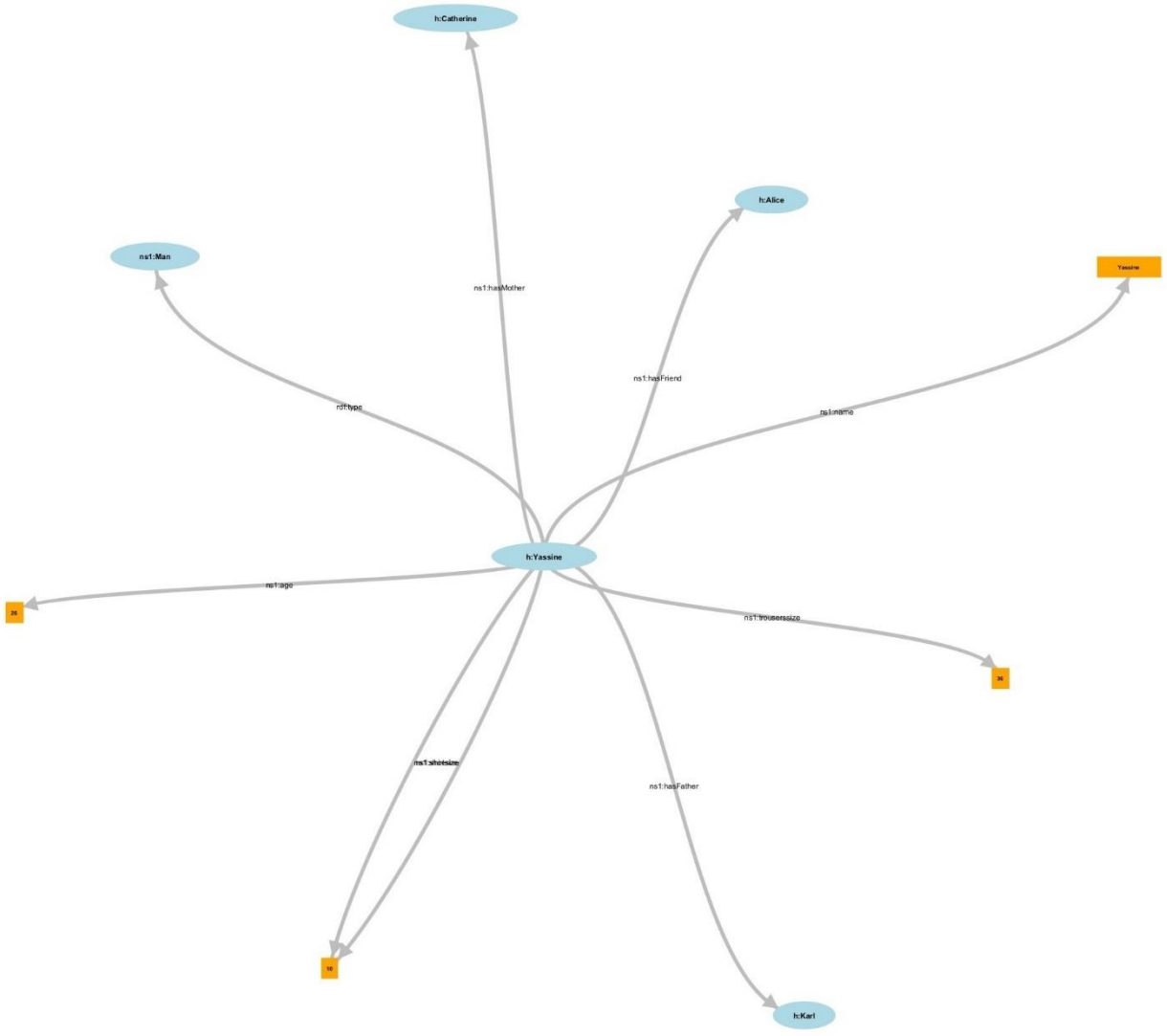
Edit the file to add your own annotation (about you) to the RDF file reusing the properties of the file. Build queries to verify and visualize the annotations you added.

Addition to file:

```
<Man rdf:ID="Yassine">
        <shoesize rdf:datatype="&xsd;integer">10</shoesize>
        <trouserssize rdf:datatype="&xsd;integer">36</trouserssize>
        <age rdf:datatype="&xsd;integer">26</age>
        <shirtsize rdf:datatype="&xsd;integer">10</shirtsize>
        <name>Yassine</name>
        <hasMother rdf:resource="#Catherine"/>
        <hasFather rdf:resource="#Karl"/>
        <hasFriend rdf:resource="#Alice"/>
</Man>
```

screenshots:

num	?p	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs#age>	26
2	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFather>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>
3	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>
4	<http://www.inria.fr/2007/09/11/humans.rdfs#hasMother>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
5	<http://www.inria.fr/2007/09/11/humans.rdfs#name>	Yassine
6	<http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize>	10
7	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	10
8	<http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize>	36
9	rdf:type	<http://www.inria.fr/2007/09/11/humans.rdfs#Man>



Question 11

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE {  ?x h:shirtsize ?size. ?y h:shirtsize ?size. FILTER (?x != ?y && ?x<?y) }
```

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT { ?x rdfs:seeAlso ?y }
WHERE {  ?x h:shirtsize ?size. ?y h:shirtsize ?size. FILTER (?x != ?y && ?x<?y) }
```

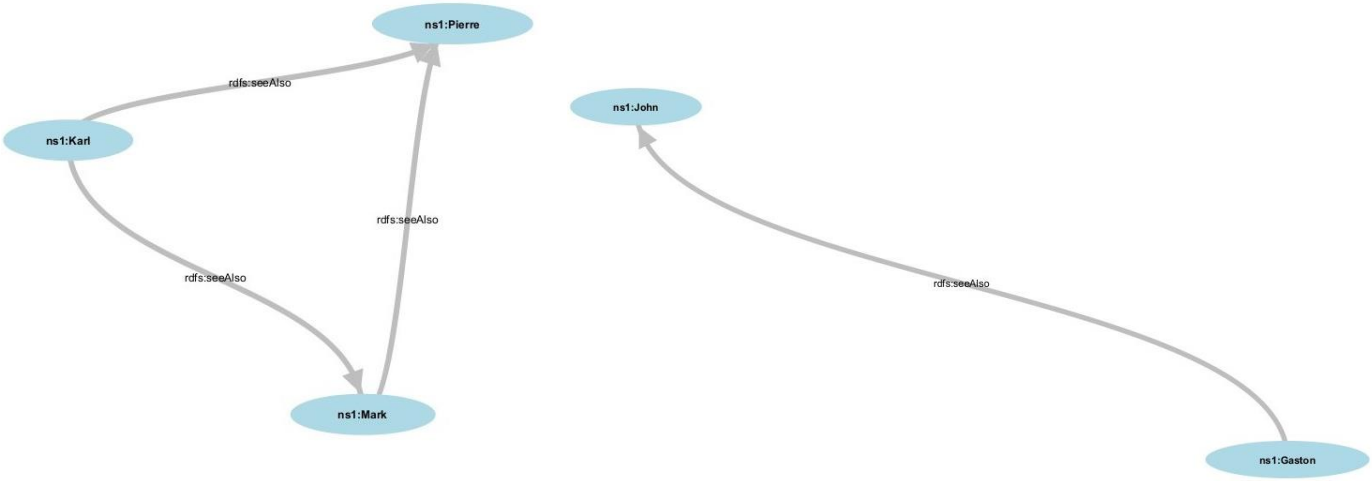
3. Change the query into an insert.

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
INSERT { ?x rdfs:seeAlso ?y }
WHERE {  ?x h:shirtsize ?size. ?y h:shirtsize ?size. FILTER (?x != ?y && ?x<?y) }
```

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

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT { ?x rdfs:seeAlso ?y }
WHERE { ?x rdfs:seeAlso ?y }
```

screenshot:



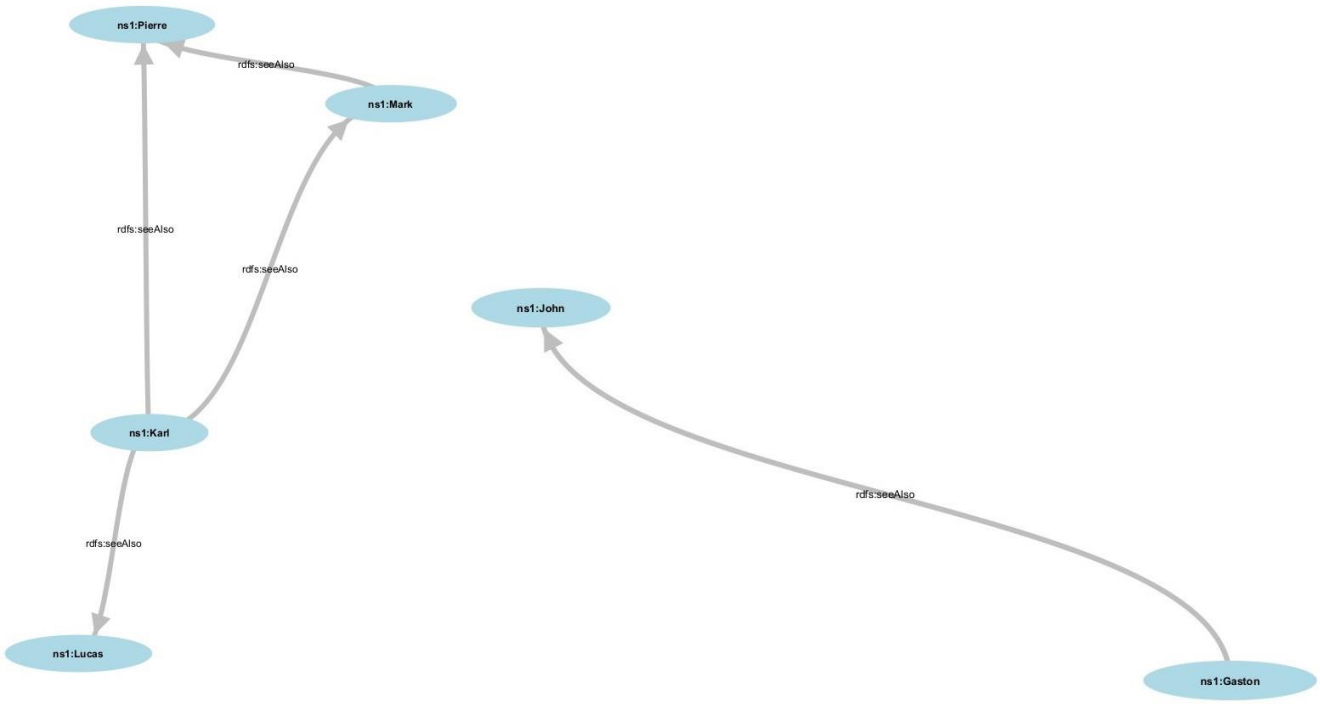
5. Adapt the first query to find persons who have the same shoe size and insert a seeAlso relationship between them.

   Query:

   PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   INSERT { ?x rdfs:seeAlso ?y }
   WHERE { ?x h:shoesize ?size. ?y h:shoesize ?size. FILTER (?x != ?y && ?x<?y) }

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 rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT { ?x rdfs:seeAlso ?y }
WHERE { ?x rdfs:seeAlso+ ?y }

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

There's nothing left as all inserted relations have been lost.

Question 12

1. Find the largest shoe size

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT (max(?size) AS ?maxshoesize)
WHERE { ?x h:shoesize ?size }

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE { {
        SELECT ( max(?size) AS ?maxshoesize )
        WHERE {  ?x h:shoesize ?size } }
        ?x h:shoesize ?maxshoesize }
```

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ( avg(?size) AS ?avgshoesize )
WHERE {  ?x h:shoesize ?size }
```

Result = **"9.285714285714286"^^xsd:decimal**

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?avgshoesize
WHERE { {
        SELECT ( sum(?size) AS ?sumshoesize ) (count(?size) AS ?countshoesize)
        WHERE {  ?x h:shoesize ?size } }
        BIND ( ?sumshoesize/?countshoesize AS ?avgshoesize ) }
```

Result = **"9.285714285714286"^^xsd:decimal**

They're the same !

Question 13

Find couples without children

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE {  ?x h:hasSpouse|^h:hasSpouse ?y. MINUS { ?x h:hasChild ?z } }
```

Question 14

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

Insert:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX i: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
INSERT DATA {  i:Yassine a h:Man;
                h:age 26;
                h:name "Yassine";
```

h:shoesize 10;
h:hasFriend i:Harry, i:Sophie. }

Screenshot result:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX i: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
SELECT *
WHERE { i:Yassine ?p ?v }

num	?p	?v
1	<http://www.inria.fr/2007/09/11/humans.rdfs#age>	26
2	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
3	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
4	<http://www.inria.fr/2007/09/11/humans.rdfs#name>	Yassine
5	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>	10
6	rdf:type	<http://www.inria.fr/2007/09/11/humans.rdfs#Man>

Question 15

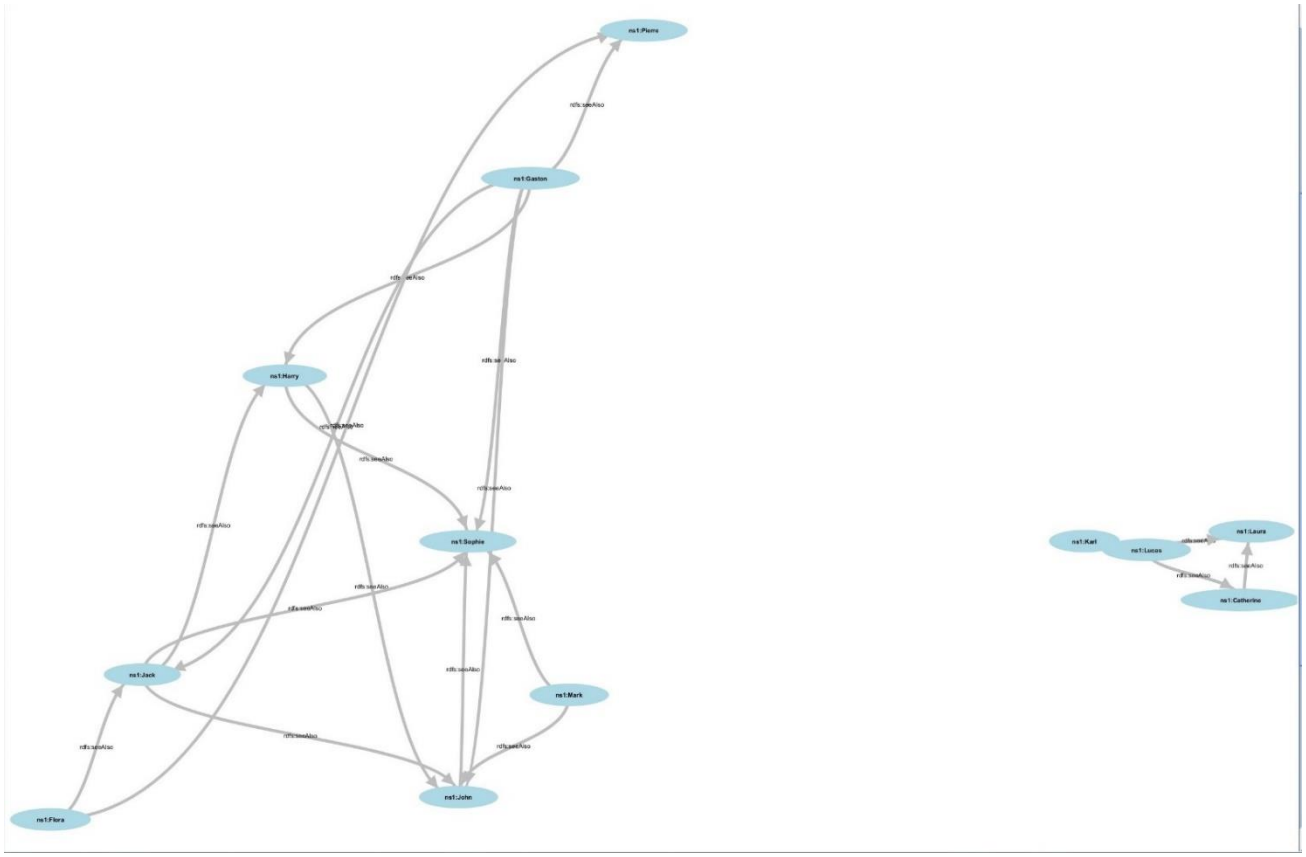
Find the people connected by paths of any family links. Construct an arc seeAlso between them to visualize the result.

query:

If we consider that spouses are also family members:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE { { ?x (h:hasChild|h:hasParent|h:hasMother|h:hasFather)+ ?y }
UNION { ?x h:hasSpouse/h:hasChild ?y }
UNION { ?x h:hasSpouse/^h:hasSpouse ?y }
FILTER (?x != ?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://www.inria.fr/2007/09/11/humans.rdfs#>
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

The Query constructs the RDF triples by using the top 20 rows of the results obtained from a query that is sent remotely to the dbpedia SPARQL endpoint. This query extracts spouses and their names using dbpedia and foaf ontologies and constructs them locally using humans.rdfs ontology.

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://www.inria.fr/2007/09/11/humans.rdfs#>
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
 }
}
```

Day 04: Answers to the practical 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: https://project.inria.fr/corese/

Create your own schema Family.rdfs

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

Your schema:

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://dsti.institute/family.rdfs> .

<#Person> a :Class ;
        :label  "human"@en, "person"@en,"homme"@fr, "personne"@fr, "être humain"@fr ;
        :comment "a member of the human species"@en, "un membre de l'espèce humaine."@fr.

<#Man> a :Class ;
        :subClassOf <#Person>;
        :seeAlso <#Woman>;
        :label "man"@en, "homme"@fr;
        :comment "an adult male person"@en, "mâle adulte de l'espèce humaine."@fr.

<#Woman> a :Class ;
        :subClassOf <#Person>;
        :seeAlso <#Man>;
        :label "woman"@en, "femme"@fr;
        :comment "an adult female person"@en, "femelle adulte de l'espèce humaine."@fr.

<#Engineer> a :Class ;
        :subClassOf <#Person>;
        :label "engineer"@en, "ingénieur"@fr;
        :comment "a person who designs or maintains machines or structures."@en, "personne qui conçoit
ou entretient des machines ou des structures."@fr.

<#Researcher> a :Class ;
        :subClassOf <#Person>;
        :label "researcher"@en, "scientist"@en, "chercheur"@fr, "scientifique"@fr;
        :comment "a person who devotes himself to doing research"@en, "personne adonnée à des
recherches spécialisées."@fr.
```

```
<#name> a rdf:Property ;
        :range :Literal;
         :label "name"@en, "nom"@fr ;
        :comment "designation of something."@en, "désignation de quelque chose."@fr .

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

<#shirtsize> a rdf:Property ;
        :domain <#Person> ;
        :label "shirt size"@en, "size"@en, "taille"@fr,"taille de chemise"@fr ;
        :comment "express in some way the approximate dimensions of the shirts of a person."@en,
"dimensions approximatives des chemises portées par une personne."@fr .

<#shoesize> a rdf:Property ;
        :domain <#Person> ;
        :label "shoe size"@en, "size"@en, "pointure"@fr ;
         :comment "express in some way the approximate length of the shoes for a person."@en, "taille,
exprimée en points, des chaussures d'une personne."@fr .

<#trouserssize> a rdf:Property ;
        :domain <#Person> ;
        :label "size"@en, "trousers size"@en, "taille"@fr, "taille de pantalon"@fr ;
         :comment "express in some way the approximate dimensions of the trousers of a person."@en,
"dimensions approximatives des pantalons portés par une personne."@fr .

<#hasChild> a rdf:Property ;
        :domain <#Person> ;
        :range <#Person> ;
         :label "has for child"@en, "a pour enfant"@fr ;
        :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 donné naissance."@fr .

<#hasFather> a rdf:Property ;
        :subPropertyOf <#hasParent>;
        :domain <#Person> ;
        :range <#Man> ;
        :label "has for father"@en, "a pour père"@fr ;
        :comment "to have for parent a male."@en, "avoir pour parent un mâle."@fr .

<#hasMother> a rdf:Property ;
        :subPropertyOf <#hasParent>;
        :domain <#Person> ;
        :range <#Woman> ;
        :label "has for mother"@en, "a pour mère"@fr ;
        :comment "to have for parent a female."@en, "avoir pour parent un femmelle."@fr .

<#hasParent> a rdf:Property ;
        :domain <#Person> ;
        :range <#Person> .
```

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

<#hasColleague> a rdf:Property ;
:domain <#Person> ;
:range <#Person> ;
:label "has for colleague"@en, "a pour collègue"@fr ;
:comment "to have a person with whom one works in a profession."@en, "avoir une personne avec qui on travaille dans une profession."@fr .

- Check that your RDF schema and RDF files are valid using the W3C's RDF validation service.
- Launch the standalone interface of Corese and load your files Family.rdfs and Jen.rdf
- 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	?t
1	rdfs:subClassOf	rdf:Property
2	rdfs:label	rdf:Property
3	rdf:type	rdf:Property
4	<http://dsti.institute/Family.rdfs-instances#Jen>	<http://dsti.institute/Family.rdfs#Woman>
5	<http://dsti.institute/Family.rdfs-instances#Jen>	<http://dsti.institute/Family.rdfs#Engineer>
6	<http://dsti.institute/Family.rdfs#name>	rdf:Property
7	<http://dsti.institute/Family.rdfs#age>	rdf:Property
8	<http://dsti.institute/Family.rdfs#shoesize>	rdf:Property
9	<http://dsti.institute/Family.rdfs#trousersize>	rdf:Property
10	<http://dsti.institute/Family.rdfs#hasSpouse>	rdf:Property
11	<http://dsti.institute/Family.rdfs-instances#Seb>	<http://dsti.institute/Family.rdfs#Man>
12	<http://dsti.institute/Family.rdfs#hasChild>	rdf:Property
13	<http://dsti.institute/Family.rdfs-instances#Anny>	<http://dsti.institute/Family.rdfs#Woman>
14	<http://dsti.institute/Family.rdfs-instances#Steffen>	<http://dsti.institute/Family.rdfs#Man>
15	<http://dsti.institute/Family.rdfs#hasFather>	rdf:Property
16	<http://dsti.institute/Family.rdfs-instances#Thomas>	<http://dsti.institute/Family.rdfs#Man>
17	<http://dsti.institute/Family.rdfs#hasColleague>	rdf:Property
18	<http://dsti.institute/Family.rdfs-instances#Fabien>	<http://dsti.institute/Family.rdfs#Man>
19	<http://dsti.institute/Family.rdfs-instances#Catherine>	<http://dsti.institute/Family.rdfs#Woman>
20	<http://dsti.institute/family.rdfs#Researcher>	rdfs:Class
21	rdfs:comment	rdf:Property
22	<http://dsti.institute/family.rdfs#Person>	rdfs:Class
23	<http://dsti.institute/family.rdfs#trouserssize>	rdf:Property
24	rdfs:domain	rdf:Property
25	<http://dsti.institute/family.rdfs#hasFather>	rdf:Property
26	rdfs:subPropertyOf	rdf:Property
27	<http://dsti.institute/family.rdfs#hasParent>	rdf:Property
28	rdfs:range	rdf:Property
29	<http://dsti.institute/family.rdfs#Man>	rdfs:Class
30	<http://dsti.institute/family.rdfs#hasChild>	rdf:Property
31	<http://dsti.institute/family.rdfs#shirtsize>	rdf:Property
32	rdfs:seeAlso	rdf:Property
33	<http://dsti.institute/family.rdfs#Woman>	rdfs:Class

- 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 h: <http://dsti.institute/family.rdfs#>
SELECT ?x
WHERE {?x a h:Human}

num	
1	<http://dsti.institute/family.rdfs-instances#Jen>
2	<http://dsti.institute/family.rdfs-instances#Seb>
3	<http://dsti.institute/family.rdfs-instances#Anny>
4	<http://dsti.institute/family.rdfs-instances#Steffen>
5	<http://dsti.institute/family.rdfs-instances#Thomas>
6	<http://dsti.institute/family.rdfs-instances#Fabien>
7	<http://dsti.institute/family.rdfs-instances#Catherine>

Explanation:

Since Man and Woman are subclasses of Human, all their instances are inferred to be Human as well.

- 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 h: <http://dsti.institute/family.rdfs#>
SELECT ?x ?y
WHERE {?x h:familyLink ?y}

num	?x	?y
1	<http://dsti.institute/family.rdfs-instances#Jen>	<http://dsti.institute/family.rdfs-instances#Seb>
2	<http://dsti.institute/family.rdfs-instances#Jen>	<http://dsti.institute/family.rdfs-instances#Anny>
3	<http://dsti.institute/family.rdfs-instances#Jen>	<http://dsti.institute/family.rdfs-instances#Steffen>
4	<http://dsti.institute/family.rdfs-instances#Seb>	<http://dsti.institute/family.rdfs-instances#Jen>
5	<http://dsti.institute/family.rdfs-instances#Seb>	<http://dsti.institute/family.rdfs-instances#Anny>
6	<http://dsti.institute/family.rdfs-instances#Seb>	<http://dsti.institute/family.rdfs-instances#Steffen>

Explanation:

Since hasChild and hasSpouse are subproperties of familyLink, all the instances related using these relations are also inferred to be related with 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:

num	
1	<http://dsti.institute/family.rdfs-instances#Jen>
2	<http://dsti.institute/family.rdfs-instances#Seb>
3	<http://dsti.institute/family.rdfs-instances#Anny>
4	<http://dsti.institute/family.rdfs-instances#Steffen>

Explanation:

Since we applied the signature on familyLink, all instances involved will be inferred as a FamilyMember.

About the human.rdfs schema

1. If you don't have the human schema file yet, download the RDF schema available at this address and save it as "human.rdfs":
   http://wimmics.inria.fr/doc/tutorial/human_2013.rdfs

2. What is the namespace associated with this ontology? How was it associated?

The namespace : the base which corresponds to : "http://www.inria.fr/2007/09/11/humans.rdfs"

It was associated through the BASE

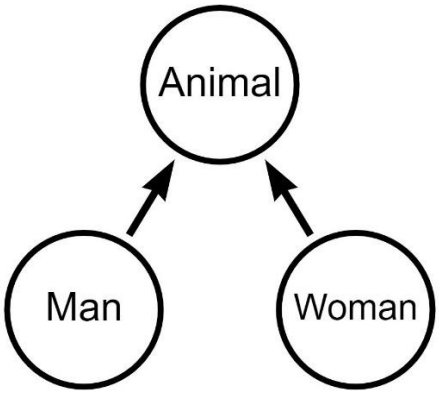
3. Look at the XML structure of this file and locate different syntactic properties: the different possible uses of the markup (ex: opening tag and closing, single tag), the use of namespaces for qualified names, the use of entities, etc.
4. Locate the use of the terms of the RDF (S) language: Class, Property, label, comment, range, domain, subClassOf, subPropertyOf, etc. To what namespaces are they associated?

Apart from "Property" which belongs to the namespace rdf, all the other statements belong to rdfs : "http://www.w3.org/2000/01/rdf-schema#"

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

There's no restriction and no implicit inference with domain and range. So all resources can have the age property.

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



Query the schema itself

Reset or relaunch the standalone Corese search engine interface and load the file `human`**`.rdfs`** (and only this one).

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

   query:

   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 rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   SELECT *
   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 h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   SELECT *
   WHERE { { h:shoesize rdfs:label ?y }
   UNION { h:shoesize rdfs:comment ?y} }

   answers:

num	?y
1	"shoe size"@en
2	"size"@en
3	"pointure"@fr
4	"express in some way the approximate length of the shoes for a person."@en
5	"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 h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   SELECT ?y
   WHERE { h:Person rdfs:label ?y. FILTER(lang(?y)="fr" && ?y!="personne"@fr) }

   answers:

num	
1	"homme"@fr
2	"Aªtre humain"@fr
3	"humain"@fr

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: To get only **English** meanings

   PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   SELECT ?y
   WHERE { ?x rdfs:comment ?y; rdfs:label "size"@en. FILTER(lang(?y)="en") }

   answers:

num	?y
1	"express in some way the approximate length of the shoes for a person."@en
2	"express in some way the approximate dimensions of the shirts of a person."@en
3	"express in some way the approximate dimensions of the trousers of a person."@en

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

   query:

   PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   SELECT DISTINCT ?x
   WHERE { ?x a rdf:Property; rdfs:domain|rdfs:range h:Person }

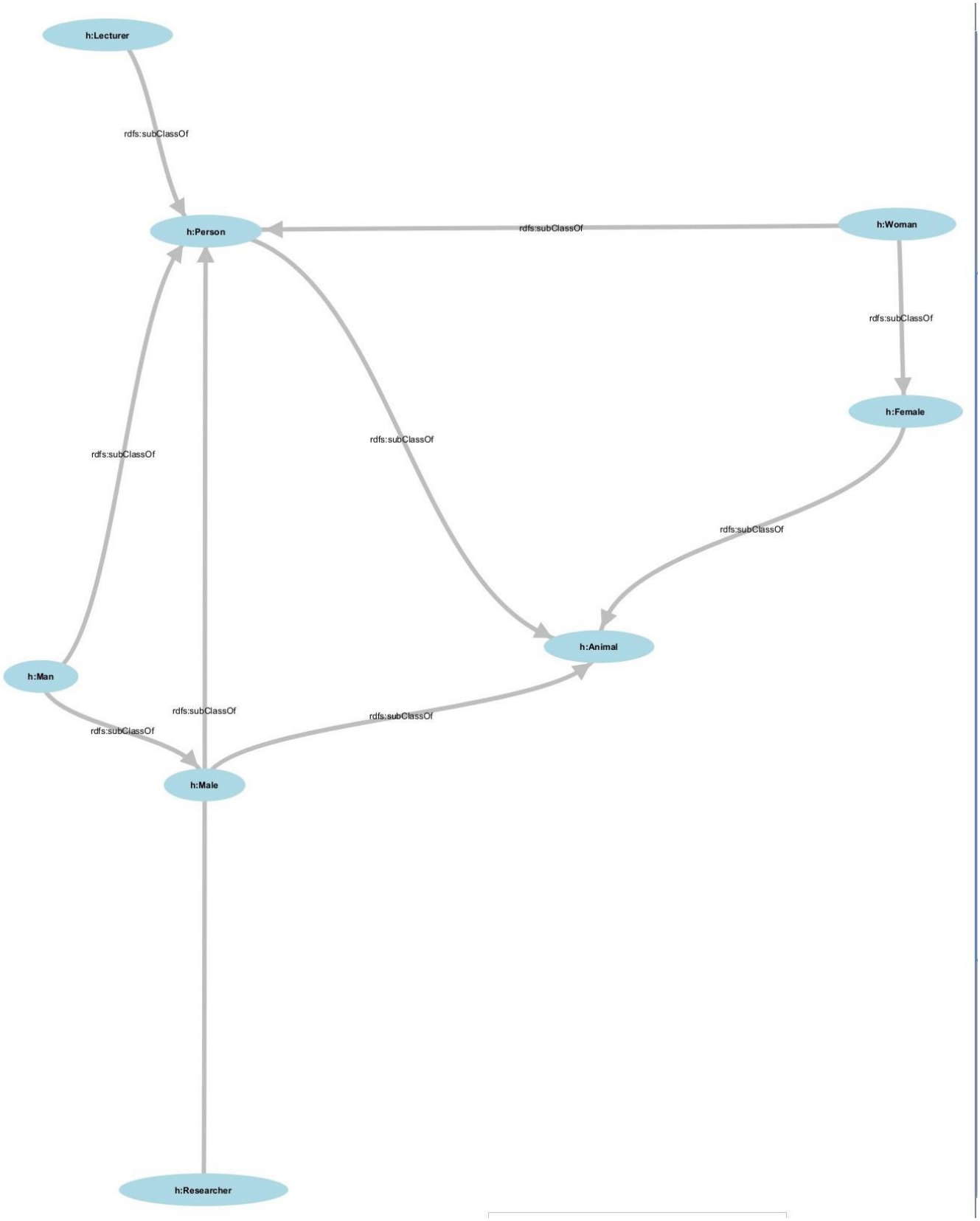
num	
1	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>
2	<http://www.inria.fr/2007/09/11/humans.rdfs#shoesize>
3	<http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize>
4	<http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize>
5	<http://www.inria.fr/2007/09/11/humans.rdfs#hasSpouse>

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

   query:

   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   CONSTRUCT { ?x rdfs:subClassOf ?y }
   WHERE { ?x rdfs:subClassOf ?y }

   screenshot:

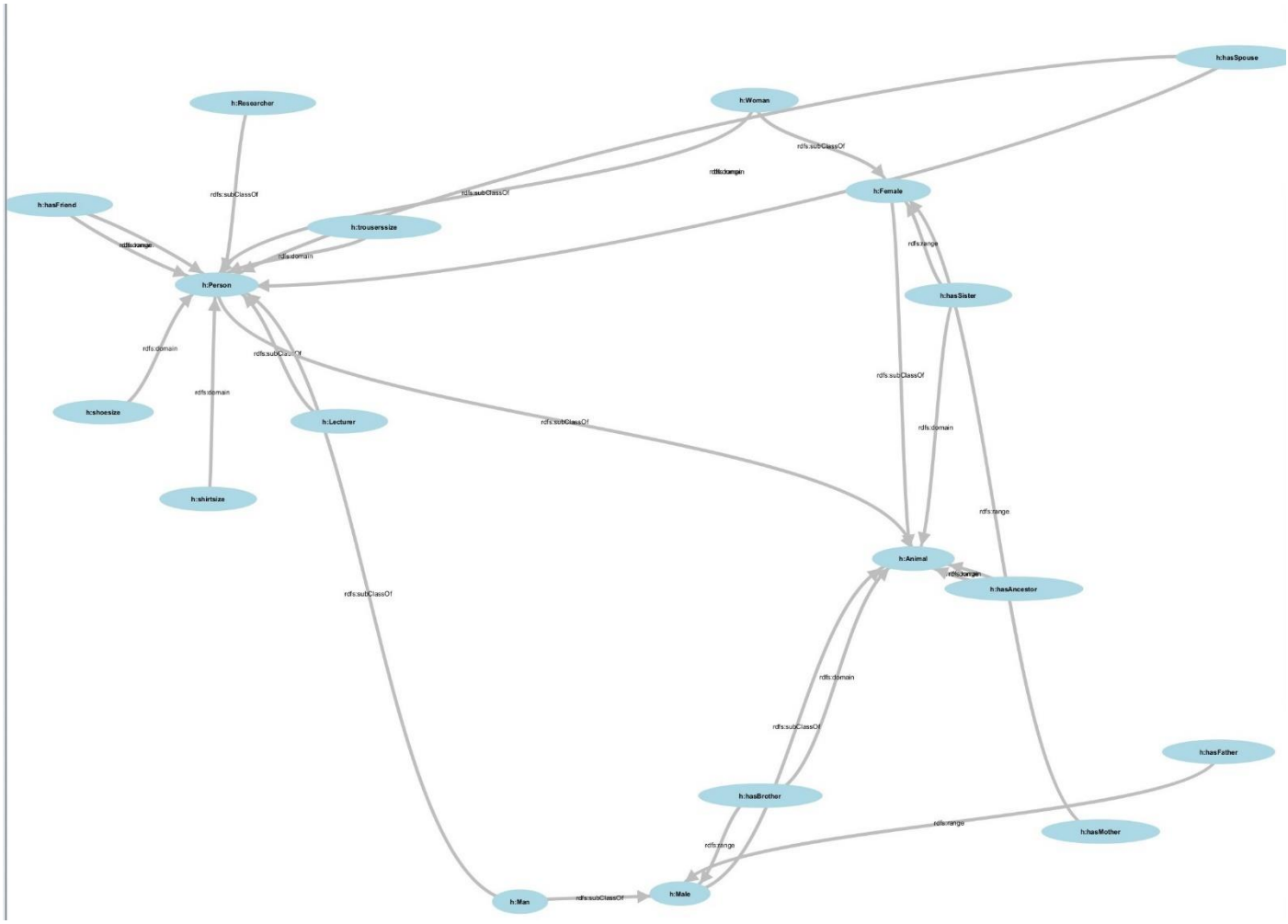


8. To the previous CONSTRUCT add the signatures of the relations.

query:

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
CONSTRUCT { ?x rdfs:subClassOf ?y. ?v rdfs:domain ?w. ?m rdfs:range?n }
WHERE { {?x rdfs:subClassOf ?y} UNION {?v rdfs:domain ?w} UNION {?m rdfs:range?n} }

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 annotations (.rdf)
2. Write a query to find the Persons.

Query:
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE { ?x a h:Person }

Number of results before: **7**

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

New number of results after and your explanation: **17. The number has grown because when the schema was added, subsumptions were added such as subclasses of Person or signatures of properties, so new entities are added to results.**

Question 2

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

Query:
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE { ?x a h:Male; h:hasSpouse ?y }

Number of results and explanation: **1 result**
The result belongs to Harry who is a Man, and the class Man is a subclass of Male, (unlike the class Person).

2. In the data declare that Lucas has as 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:
Karl hasSpouse Catherine

Number of results before and after and explanation: **2**
This line has been added because hasFather Property has a range Male, which is applied to Karl when we create the relation. Being a Male, he appears in the results of the query.

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 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE { ?x a h:Lecturer; rdf:type ?y }

Number of results and your explanation: **7 results**

Eve is a Person and Lecturer because she was declared as so.

Laura is a Person, Lecturer and Research because she was declared as so, but were also added Female and Animal. Animal comes from the fact Person is subclass of Animal, and Female from the fact she was assigned as a Mother of Catherine.

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

Query:
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE { ?x a h:Person, h:Male }

Your explanation of the result:
We get Jack because he is a Man, which is a subclass of both Male and Person.

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 h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE { ?x h:hasAncestor ?y  }
```

Your explanation of the result:
We get 5 results which come from the fact they were defined as hasMother and hasFather, 2 properties that are sub-properties of hasParent which is itself a sub-property of hasAncestor.

Question 5

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y ?child
WHERE { ?x h:hasSpouse|^h:hasSpouse ?y.
OPTIONAL { ?x h:hasChild|^h:hasParent ?child } }
```

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

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
CONSTRUCT { ?x h:hasSpouse ?y. ?child h:hasParent ?x}
WHERE { ?x h:hasSpouse|^h:hasSpouse ?y.
OPTIONAL { ?x h:hasChild|^h:hasParent ?child } }
```

Question 6

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

Addition to schema:

```
<rdf:Property rdf:ID="olderThan">
  <domain rdf:resource="#Person"/>
  <range rdf:resource="#Person"/>
  <label xml:lang="en">is older Than</label>
```

```
  <comment xml:lang="en">relation between a person and another person that describes who is
older.</comment>
  <label xml:lang="fr">plus âgé que</label>
  <comment xml:lang="fr">relation entre une personne et une autre personne qui décrit qui est plus âgé que
l'autre.</comment>
</rdf:Property>
```

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
CONSTRUCT { ?x h:olderThan ?y }
WHERE {?x h:age ?xage. ?y h:age ?yage. FILTER ( ?xage > ?yage ) }
```

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

Query:

```
PREFIX h:<http://www.inria.fr/2007/09/11/humans.rdfs#> .
CONSTRUCT { ?x h:olderThan ?y}
WHERE { ?x h:age ?xage.
        ?y h:age ?yage.
        FILTER (?xage > ?yage && NOT EXISTS { z h:age ?zage.  FILTER (?xage > ?zage && ?zage > ?yage) } ) }
```

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 i: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
SELECT DISTINCT ?p ?v
WHERE {i:John ?p ?v. ?p 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:

My query returns all the French labels and comments of both types and properties associated with Laura.

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX i: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
SELECT ?entity ?label ?comment
WHERE { { i:Laura rdf:type ?type.
          ?type rdfs:label ?label; rdfs:comment ?comment.
          FILTER( lang(?label)="fr" &&  lang(?comment)="fr"  )
          BIND (STRAFTER(?type, STR(h:)) AS ?entity)}
          UNION
         { i:Laura ?prop ?value.
```

```
        ?prop rdfs:label ?label; rdfs:comment ?comment.
        FILTER( lang(?label)="fr" &&  lang(?comment)="fr"  )
        BIND (STRAFTER(?prop, STR(h:)) AS ?entity) }
}
```

?entity	?label	?comment
Person	"homme"@fr	"un membre de l'espA¨ce humaine."@fr
Person	"personne"@fr	"un membre de l'espA¨ce humaine."@fr
Person	"Aªtre humain"@fr	"un membre de l'espA¨ce humaine."@fr
Person	"humain"@fr	"un membre de l'espA¨ce humaine."@fr
Lecturer	"professeur"@fr	"personne qui enseigne une discipline, une technique, un art."@fr
Researcher	"chercheur"@fr	"personne adonnA©e A des recherches spA©cialisA©es."@fr
Researcher	"scientifique"@fr	"personne adonnA©e A des recherches spA©cialisA©es."@fr
Animal	"animal"@fr	"Aªtre vivant douA© de sensibilite de mobilitA©."@fr
Female	"femelle"@fr	"animal appartenant au sexe apte A produire des ovules."@fr
hasFriend	"a pour ami"@fr	"relation entre une personne et une autre personne qui est l'objet d'un attachement privilA©giA©."@fr
name	"nom"@fr	"dA©signation de quelque chose."@fr

Day 04: Answers to the practical 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: https://project.inria.fr/corese/

A, Query data augmented by an OWL schema

Make a copy of the human.rdfs file, name it humans.owl 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.rdfs or the humans.owl you modified.

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

Code added to the schema:

<owl:SymmetricProperty rdf:ID="hasSpouse"/>
<owl:SymmetricProperty rdf:ID="hasFriend"/>

Query: (*the same can be done for h:hasFriend*)

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE {?x h:hasSpouse ?y}

Result before addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Result after addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>
11	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
12	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Explanation:

Since hasSpouse has become symmetric, this relation once applied in one direction, becomes also applied in the other. So we get the same results as when querying with : WHERE {?x h:hasSpouse|^h:hasSpouse ?y}

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

Code added to the schema:

<rdf:Property rdf:ID="hasChild">
<owl:inverseOf rdf:resource="#hasParent"/>
</rdf:Property>

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE {?x h:hasChild ?y}

Result before addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>

Result after addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>

Explanation:

Since hasChild is the invere of hasParent, all resources linked in one direction with hasParent will also be assigned the inverse property hasChild in the other direction.

3. Declare `hasAncestor` as transitive property.

Code added to the schema:

<owl:TransitiveProperty rdf:ID="hasAncestor"/>

Query:

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE {?x h:hasAncestor ?y}

Result before addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Result after addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>
9	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie>
10	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
11	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>
12	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
13	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
14	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora>
15	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>
16	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>
17	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl>
18	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Explanation:

Since hasAncestor is transitive, it propagates from peers to peers. For example, in the first result, we see that Harry has as ancestor Jack (line 1) and Jack has as ancestor Gaston (line 5). With transitivity, a link will be added between Harry and Gaston, as Gaston now is declared as an ancestor of Harry, which we can see in the second result (line 1).

4. 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:

```
<Class rdf:ID="Male">
  <owl:disjointWith rdf:resource="#Female"/>
  …
</Class>
```

Query:

```
PREFIX h:<http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?p WHERE { ?x ?p ?y. ?y h:name "Jack"}
```

Result before addition to the schema:

num	?x	?p
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasChild>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasParent>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	owl:sameAs

Result after addition to the schema:

num	?x	?p
1	:b_41.0.10.44.42	sp:violationRoot
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasAncestor>
5	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasChild>
6	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend>
7	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry>	<http://www.inria.fr/2007/09/11/humans.rdfs#hasParent>
8	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	owl:sameAs

num	?x	?y	?p
1	:b_41.0.10.44.42	<http://www.inria.fr/2007/09/11/humans.rdfs#Male>	sp:arg1
2	:b_41.0.10.44.42	<http://www.inria.fr/2007/09/11/humans.rdfs#Female>	sp:arg2
3	:b_41.0.10.44.42	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack>	sp:violationRoot

Explanation:

A violation is added to the results, which is due to the fact Jack is declared as both male and female whilst male and female are disjoint classes.

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

Code added to the schema:

```
<owl:Class rdf:ID="Professor">
        <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Researcher"/>
                <owl:Class rdf:about="#Lecturer"/>
        </owl:intersectionOf>
</owl:Class>
```

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE {?x a h:Professor}
```

Result before addition to the schema:

None

Result after addition to the schema:

num	?x
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Explanation:

Laura is declared as both a Lecturer and Researcher, and since Professor is the intersection of both, Laura is also declared as Professor.

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

Code added to the schema:

```
<owl:Class rdf:ID="Academic">
```

```
        <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Researcher"/>
                <owl:Class rdf:about="#Lecturer"/>
        </owl:unionOf>
</owl:Class>
```

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x
WHERE {?x a h:Academic}
```

Result before addition to the schema:

None

Result after addition to the schema:

num	?x
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#David>
3	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston>
4	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura>

Explanation:

Each resource of the 4 results belong either to Researcher or Lecturer type, and since Academic is the union of both, they're all declared to be Academic as well.

7. 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):

<u>New classes and property:</u>

```
<Class rdf:ID="Organization">
        <subClassOf rdf:resource="#Organization"/>
        <label xml:lang="en">organization</label>
        <label xml:lang="fr">organisation</label>
</Class>

<Class rdf:ID="University">
        <subClassOf rdf:resource="#Organization"/>
        <label xml:lang="en">university</label>
        <label xml:lang="fr">université</label>
</Class>

<rdf:Property rdf:ID="mainEmployer">
        <domain rdf:resource="#Person"/>
        <range rdf:resource="#Organization"/>
```

```
        <label xml:lang="en">has for main employer</label>
        <label xml:lang="fr">a pour principal employeur</label>
</rdf:Property>
```

New classes and restriction:

```
<owl:Class rdf:ID="Professor">
        <subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#mainEmployer" />
                        <owl:allValuesFrom rdf:resource="#University" />
                </owl:Restriction>
        </subClassOf>
        <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Researcher"/>
                <owl:Class rdf:about="#Lecturer"/>
        </owl:intersectionOf>
</owl:Class>
```

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

```
<Organization rdf:about="#DSTI"/>

<Researcher rdf:about="#Laura">
  <name>Laura</name>
  <mainEmployer rdf:resource="#DSTI"/>
</Researcher>
```

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?employer ?type
WHERE { ?x a h:Professor; h:mainEmployer ?employer. OPTIONAL { ?employer a ?type } }
```

Result before addition to the schema:

num	?employer	?type
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#DSTI>	<http://www.inria.fr/2007/09/11/humans.rdfs#Organization>

Result after addition to the schema:

n...	?employer	?type
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#DSTI>	<http://www.inria.fr/2007/09/11/humans.rdfs#Organization>
2	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#DSTI>	<http://www.inria.fr/2007/09/11/humans.rdfs#University>

Explanation:

In the first result, Professor Laura is declared to has Organization DSTI as a main employer, so we get DSTI being of type Organization only. When we added the restriction, all Professor instances have their mainEmployer property restricted to University. Consequently, Organization DSTI is inferred to be a University in the second result.

8. 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:

```
<Class rdf:ID="Person">
        <owl:equivalentClass>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasParent" />
                        <owl:someValuesFrom rdf:resource="#Woman" />
                </owl:Restriction>
        </owl:equivalentClass>
        ….
        ….
</Class>
```

Code added to the data (Adding an animal that has a Woman parent):

```
<Animal rdf:ID="Milo">
        <hasParent rdf:resource="#Alice"/>
</Animal>
```

Query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT ?x ?y
WHERE { ?x h:hasParent ?y. FILTER ( not exists { ?x a h:Person } ) }
```

Result before addition to the schema:

num	?x	?y
1	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Milo>	<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice>

Result after addition to the schema:

num	?x	?y

Explanation:

Since a Person is restricted to an instance that must have a Woman as a parent, any instance that satisfies this condition is inferred to become a Person. Which is why the animal Milo, who has the Woman Alice as a parent, became also a Person instance, and wasn't shown in the second result.

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.

OWL Primitive	**Example Meaning**	**Example OWL definition**
owl:oneOf	Days of a week.	<owl:Class rdf:id="DayOfWeek"> <owl:oneOf rdf:parseType="Collection"> <owl:Thing rdf:ID="Monday"/> <owl:Thing rdf:ID="Tuesday"/> <owl:Thing rdf:ID="Wednesday"/> <owl:Thing rdf:ID="Thursday"/> <owl:Thing rdf:ID="Friday"/> <owl:Thing rdf:ID="Saturday"/> <owl:Thing rdf:ID="Sunday"/> </owl:oneOf> </owl:Class>
owl:unionOf	Parents can be a father or a mother.	<owl:Class rdf:id="Parent"> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:ID="Mother"/> <owl:Class rdf:ID="Father"/> </owl:unionOf> </owl:Class>
owl:intersectionOf	A bachelor is an unmarried man.	<owl:Class rdf:id="Bachelor"> <owl:intersectionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Man"/> <owl:Class rdf:about="#Unmarried"/> </owl:intersectionOf> </owl:Class>
owl:complementOf	A person is either adult or minor.	<owl:Class rdf:ID="Adult"> <owl:complementOf rdf:resource="#Minor"/> </owl:Class>
owl:disjointUnionOf	The four Seasons form a disjoint union.	<owl:Class rdf:about="Season"> <owl:disjointUnionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Winter"/> <owl:Class rdf:about="#Spring"/> <owl:Class rdf:about="#Summer"/> <owl:Class rdf:about="#Autumn"/> </owl:disjointUnionOf> </owl:Class>
owl:ObjectProperty	We can define the property of a person working at a subsidiary of a company	<owl:ObjectProperty rdf:about="worksAt"> <rdfs:domain rdf:resource="Person"/> <rdfs:range rdf:resource="Subsidiary"/> </owl:ObjectProperty>
owl:DatatypeProperty	A person's or company's email	<owl:DatatypeProperty rdf:ID="email"/> <rdfs:range rdf:resource="&xsd;string"/> </owl:DatatypeProperty>
owl:SymmetricProperty	Synonyms are synonyms of each other.	<owl:SymmetricProperty rdf:ID="synonymOf"/>
owl:inverseOf	Teacher and student relationship	<rdf:Property rdf:ID="hasTeacher"> <owl:inverseOf rdf:resource="#hasStudent" /> </rdf:Property>
owl:TransitiveProperty	Being richer than somebody else.	<owl:TransitiveProperty rdf:ID="isRicherThan" />

owl:propertyDisjointWith	Being friends and enemies are 2 properties that cannot coexist.	<owl:ObjectProperty rdf:about="isFriendWith"> <owl:propertyDisjointWith rdf:resource="isEnemyWith"/> </owl:ObjectProperty>
owl:IrreflexiveProperty	Antonyms, a word cannot be antonym of itself.	<owl:IrreflexiveProperty rdf:about="antonymOf"/>
owl:propertyChainAxiom	If a person is born at a country that has an official language, then the person is fluent at that language.	<owl:ObjectProperty rdf:ID="fluentAt"> <owl:propertyChainAxiom rdf:parseType="Collection"> <owl:ObjectProperty rdf:about="#bornAt"/> <owl:ObjectProperty rdf:about="#officialLanguage"/> </owl:propertyChainAxiom> </owl:ObjectProperty>
owl:FunctionalProperty	The model of a car, airplane...	<owl:FunctionalProperty rdf:ID="hasModel" />
owl:InverseFunctionalProperty	Fingerprint templates identify unique people.	<owl:InverseFunctionalProperty rdf:ID="fingerprintTemplate" />
owl:hasKey	Customers worldwide can be identified using their national ID and country.	<owl:Class rdf:ID="Customer"> <owl:hasKey rdf:parseType="Collection"> <owl:ObjectProperty rdf:about="#nationalID"/> <owl:ObjectProperty rdf:about="#country"/> </owl:hasKey> </owl:Class>
owl:allValuesFrom	A dog trainer necessarily trains dogs.	<owl:Class rdf:ID="DogTrainer"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#trains"/> <owl:allValuesFrom rdf:resource="#Dog"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
owl:someValuesFrom	A bakery makes flour-based food such as bread, pastry, pie.. Bread is an essential component that defines a bakery.	<owl:Class rdf:ID="Bakery"> <owl:equivalentClass> <owl:Restriction> <owl:onProperty rdf:resource="#makes"/> <owl:someValuesFrom rdf:resource="#Bread"/> </owl:Restriction> </owl:equivalentClass> </owl:Class>
owl:hasValue	A diet drink is a drink that is sugar free.	<owl:Class rdf:ID="DietDrink"> <rdfs:subClassOf rdf:resource="#Drink"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#sugarFree"/> <owl:hasValue rdf:datatype="&xsd;boolean">true </owl:hasValue> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
owl:maxCardinality	A person can only have 2 biological parents.	<owl:Class rdf:ID="Person"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#hasBiologicalParent"/> <owl:maxCardinality rdf:datatype="&xsd;integer">2

		</owl:maxCardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
owl:qualifiedCardinality	A competition can have only one winner.	<owl:Class rdf:ID="Competition"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#hasWinner" /> <owl:onClass rdf:resource="#Participant" /> <owl:qualifiedCardinality>1</owl:qualifiedCardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>