Semantic Web handout including: lecture questions and practical sessions

In this document, you must provide your answers to the questions asked during the course **and** to the questions of the practical sessions; everything in one document.

The questions of the course have been repeated here; **do not delete the questions** but provide your answer to each question just below the question. You can use screenshots when appropriate as an answer to a question.

At the end, you must generate and submit only one final PDF file based on this template.

In questions where you are asked to create, invent or use your own data, make sure they are different from other student's.

First name: Sophie

Family name: Gallet

Email: sophie.gallet@edu.dsti.institute

QUESTIONS FROM THE COURSES

Day 01: questions from the course.

Q1.1 Practice XML replace missing parts

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

Q1.2 Provide 10 first lines

Get 10 first lines of the five results for:

http://www.wikidata.org/entity/Q23014205
http://www.wikidata.org/entity/Q23014205.json
http://www.wikidata.org/entity/Q23014205.rdf
http://www.wikidata.org/entity/Q23014205.ttl
http://www.wikidata.org/entity/Q23014205.nt

```
<!DOCTYPE html>
<html class="client-js gr__wikidata_org ve-not-available" lang="en"
dir="ltr"><head>
<meta charset="UTF-8">
<title>Fabien Gandon - Wikidata</title>
<script>document.documentElement.className="client-
js";RLCONF={"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":!
1,"wgNamespaceNumber":
0,"wgPageName":"Q23014205","wgTitle":"Q23014205","wgCurRevisionId":
872123137,"wgRevisionId":872123137,"wgArticleId":25028548,"wgIsArticle":!
0,"wgIsRedirect":!1,"wgAction":"view","wgUserName":null,"wgUserGroups":
["*"],"wgCategories":[],"wgBreakFrames":!
1,"wgPageContentLanguage":"en","wgPageContentModel":"wikibase-
item","wgSeparatorTransformTable":["",""],"wgDigitTransformTable":
["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":
["","January","February","March","April","May","June","July","August","Septembe
r","October","November","December"],"wgMonthNamesShort":
["","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],"w
gRelevantPageName":"Q23014205","wgRelevantArticleId":
25028548,"wgRequestId":"XYiIOwpAICoAAJSZR5AAAACK","wgCSPNonce":!
```

1,"wgIsProbablyEditable":!0,"wgRelevantPageIsProbablyEditable":!
0,"wgRestrictionEdit":[

],"wgMediaViewerOnClick":!0,"wgMediaViewerEnabledByDefault":!
0,"wgVisualEditor":
{"pageLanguageCode":"en","pageLanguageDir":"ltr","pageVariantFallbacks":"en"},"
wgMFDisplayWikibaseDescriptions":{"search":!0,"nearby":!0,"watchlist":!
0,"tagline":!1},"wgWMESchemaEditAttemptStepOversample":!1,"wgPoweredByHHVM":!
1,"wgULSCurrentAutonym":"English","wgNoticeProject":"wikidata","wbIsEditView":!
0,"wbEntityId":"Q23014205","wgEditSubmitButtonLabelPublish":!
0,"wbUserSpecifiedLanguages":[],"wbCopyright":
{"version":"wikibase-1","messageHtml":"By clicking \"publish\", you agree to
the \u003Ca href=\"https://foundation.wikimedia.org/wiki/Terms_of_Use\"
class=\"extiw\" title=\"wikimedia:Terms of Use\"\u003Eterms of use\u003C/
a\u003E, and you irrevocably agree to release your contribution under the
\u003Ca rel=\"nofollow\" class=\"external text\" href=\"https://
creativecommons.org/publicdomain/zero/1.0/\"\u003ECreative Commons CC0
License\u003C/a\u003E."},"wbBadgeItems":{"Q17437798":

"wb-badge-goodarticle","Q17437796":"wb-badge-featuredarticle","Q17559452":"wb-
badge-recommendedarticle","Q17506997":"wb-badge-featuredlist","Q17580674":"wb-
badge-featuredportal","Q20748091":"wb-badge-notproofread","Q20748094":"wb-
badge-problematic","Q20748092":"wb-badge-proofread","Q20748093":"wb-badge-
validated","Q28064618":"wb-badge-digitaldocument","Q51759403":"wb-badge-
goodlist"},"wbMultiLingualStringLimit":250,"wgCentralAuthMobileDomain":!
1};RLSTATE={"ext.globalCssJs.user.styles":"ready","site.styles":"ready","noscri
pt":"ready","user.styles":"ready","ext.globalCssJs.user":"ready","user":"ready"
,"user.options":"ready","user.tokens":"loading","wikibase.common":"ready","jque
ry.ui.core.styles":"ready","jquery.wikibase.statementview.RankSelector.styles":
"ready","jquery.wikibase.toolbar.styles":"ready","jquery.wikibase.toolbarbutton
.styles":"ready","mediawiki.legacy.shared":"ready","mediawiki.legacy.commonPrin
t":"ready","jquery.makeCollapsible.styles":"ready",

"ext.visualEditor.desktopArticleTarget.noscript":"ready","ext.uls.pt":"ready","
ext.wikidata-org.badges":"ready","ext.wikimediaBadges":"ready","ext.
3d.styles":"ready","mediawiki.skinning.interface":"ready","skins.vector.styles"
:"ready"};RLPAGEMODULES=["wikibase.ui.entityViewInit","site","mediawiki.page.st
artup","mediawiki.page.ready","jquery.makeCollapsible","mediawiki.searchSuggest
","wikibase.ui.entitysearch","ext.gadget.Search","ext.gadget.AuthorityControl",
"ext.gadget.SiteIdToInterwiki","ext.gadget.ProtectionIndicators","ext.gadget.Po
pupsFix","ext.gadget.imagelinks","ext.gadget.NewSection","ext.gadget.formWizard
","ext.centralauth.centralautologin","mmv.head","mmv.bootstrap.autostart","ext.
visualEditor.desktopArticleTarget.init","ext.visualEditor.targetLoader","ext.ci
toid.wikibase.init","ext.eventLogging","ext.wikimediaEvents","ext.wikimediaEven
ts.wikibase","ext.navigationTiming","ext.uls.compactlinks","ext.uls.interface",
"propertySuggester.suggestions",

"wikibase.quality.constraints.suggestions","ext.centralNotice.geoIP","ext.centr
alNotice.startUp","skins.vector.js"];</script>

<script>(RLQ=window.RLQ||[]).push(function()
{mw.loader.implement("user.tokens@tffin",function($,jQuery,require,module){/
@nomin/mw.user.tokens.set({"editToken":"+\\","patrolToken":"+\
\","watchToken":"+\\","csrfToken":"+\\"});

JSON

{"entities":{"Q23014205":{"pageid":25028548,"ns":
0,"title":"Q23014205","lastrevid":
872123137,"modified":"2019-03-02T12:03:45Z","type":"item","id":"Q23014205","lab

els":{"fr":{"language":"fr","value":"Fabien Gandon"},"en":{"language":"en","value":"Fabien Gandon"},"br":{"language":"br","value":"Fabien Gandon"},"de":{"language":"de","value":"Fabien Gandon"},"af":{"language":"af","value":"Fabien Gandon"},"an":{"language":"an","value":"Fabien Gandon"},"ast":{"language":"ast","value":"Fabien Gandon"},"bar":{"language":"bar","value":"Fabien Gandon"},"bm":{"language":"bm","value":"Fabien Gandon"},"ca":{"language":"ca","value":"Fabien Gandon"},"co":{"language":"co","value":"Fabien Gandon"},"cs":{"language":"cs","value":"Fabien Gandon"},"cy":{"language":"cy","value":"Fabien Gandon"},"da":{"language":"da","value":"Fabien Gandon"},"de-at":{"language":"de-at","value":"Fabien Gandon"},"de-ch":{"language":"de-ch","value":"Fabien Gandon"},"en-ca":{"language":"en-ca","value":"Fabien Gandon"},"en-gb":{"language":"en-gb","value":"Fabien Gandon"},"eo":{"language":"eo","value":"Fabien Gandon"},"es":{"language":"es","value":"Fabien Gandon"},"et":{"language":"et","value":"Fabien Gandon"},"eu":{"language":"eu","value":"Fabien Gandon"},"fi":{"language":"fi","value":"Fabien Gandon"},"frc":{"language":"frc","value":"Fabien Gandon"},"frp":{"language":"frp","value":"Fabien Gandon"},"fur":{"language":"fur","value":"Fabien Gandon"},"ga":{"language":"ga","value":"Fabien Gandon"},"gd":

RDF

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:ontolex="http://www.w3.org/ns/lemon/ontolex#" xmlns:dct="http://purl.org/dc/terms/" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:wikibase="http://wikiba.se/ontology#" xmlns:skos="http://www.w3.org/2004/02/skos/core#" xmlns:schema="http://schema.org/" xmlns:cc="http://creativecommons.org/ns#" xmlns:geo="http://www.opengis.net/ont/geosparql#" xmlns:prov="http://www.w3.org/ns/prov#" xmlns:v="http://www.wikidata.org/value/" xmlns:wd="http://www.wikidata.org/entity/" xmlns:data="https://www.wikidata.org/wiki/Special:EntityData/" xmlns:s="http://www.wikidata.org/entity/statement/" xmlns:ref="http://www.wikidata.org/reference/" xmlns:wdt="http://www.wikidata.org/prop/direct/" xmlns:wdtn="http://www.wikidata.org/prop/direct-normalized/" xmlns:p="http://www.wikidata.org/prop/" xmlns:ps="http://www.wikidata.org/prop/statement/" xmlns:psv="http://www.wikidata.org/prop/statement/value/" xmlns:psn="http://www.wikidata.org/prop/statement/value-normalized/" xmlns:pq="http://www.wikidata.org/prop/qualifier/" xmlns:pqv="http://www.wikidata.org/prop/qualifier/value/" xmlns:pqn="http://www.wikidata.org/prop/qualifier/value-normalized/" xmlns:pr="http://www.wikidata.org/prop/reference/" xmlns:prv="http://www.wikidata.org/prop/reference/value/" xmlns:prn="http://www.wikidata.org/prop/reference/value-normalized/" xmlns:wdno="http://www.wikidata.org/prop/novalue/">
    <rdf:Description rdf:about="https://www.wikidata.org/wiki/Special:EntityData/Q23014205">
        <rdf:type rdf:resource="http://schema.org/Dataset"/>
        <schema:about rdf:resource="http://www.wikidata.org/entity/Q23014205"/>
        <cc:license rdf:resource="http://creativecommons.org/publicdomain/zero/1.0/"/>
        <schema:softwareVersion>1.0.0</schema:softwareVersion>

```
            <schema:version rdf:datatype="http://www.w3.org/2001/
XMLSchema#integer">872123137</schema:version>

            <schema:dateModified rdf:datatype="http://www.w3.org/2001/
XMLSchema#dateTime">2019-03-02T12:03:45Z</schema:dateModified>

            <wikibase:statements rdf:datatype="http://www.w3.org/2001/
XMLSchema#integer">29</wikibase:statements>
```

TTL

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ontolex: <http://www.w3.org/ns/lemon/ontolex#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix wikibase: <http://wikiba.se/ontology#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix schema: <http://schema.org/> .
@prefix cc: <http://creativecommons.org/ns#> .
```

NT

```
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://
www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Dataset> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://
schema.org/about> <http://www.wikidata.org/entity/Q23014205> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://
creativecommons.org/ns#license> <http://creativecommons.org/publicdomain/zero/
1.0/> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://
schema.org/softwareVersion> "1.0.0" .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://
schema.org/version> "872123137"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://
schema.org/dateModified> "2019-03-02T12:03:45Z"^^<http://www.w3.org/2001/
XMLSchema#dateTime> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://wikiba.se/
ontology#statements> "29"^^<http://www.w3.org/2001/XMLSchema#integer> .
<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://wikiba.se/
ontology#identifiers> "10"^^<http://www.w3.org/2001/XMLSchema#integer> .

<https://www.wikidata.org/wiki/Special:EntityData/Q23014205> <http://wikiba.se/
ontology#sitelinks> "0"^^<http://www.w3.org/2001/XMLSchema#integer> .

<http://www.wikidata.org/entity/Q23014205> <http://www.w3.org/1999/02/22-rdf-
syntax-ns#type> <http://wikiba.se/ontology#Item> .
```

Q1.3 DBpedia

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

```
dbp:populationDemonym : Londoner
rdf:type (as requested on the teacher's blackboard, not on this pdf) :
owl:Thing
dbo:Place
dbo:Location
wikidata:Q486972
dbo:PopulatedPlace
dbo:Settlement
geo:SpatialThing
schema:Place
umbel-rc:Location_Underspecified
umbel-rc:PopulatedPlace
umbel-rc:Village
yago:WikicatStaplePorts
yago:AdministrativeDistrict108491826
yago:Area108497294
yago:Capital108518505
yago:Center108523483
yago:City108524735
yago:District108552138
yago:GeographicPoint108578706
yago:GeographicalArea108574314
yago:Location100027167
yago:Municipality108626283
yago:Object100002684
yago:PhysicalEntity100001930
yago:Point108620061
yago:Port108633957
yago:Region108630985
yago:Seat108647945
yago:Site108651247
yago:Town108665504
yago:Tract108673395
yago:UrbanArea108675967
yago:YagoGeoEntity
yago:YagoLegalActorGeo
yago:YagoPermanentlyLocatedEntity
yago:WikicatCapitalsInEurope
yago:WikicatCitiesInEngland
yago:WikicatCitiesWithMillionsOfInhabitants
yago:WikicatArthurianLocations
yago:WikicatBritishCapitals
yago:WikicatPopulatedPlacesEstablishedInThe1stCentury
yago:WikicatPortCities
yago:WikicatPortCitiesAndTownsInEngland
yago:WikicatPortsAndHarbours
yago:WikicatPostTownsInPostcodeAreasCoveringLondon

Vienna URI: http://dbpedia.org/resource/Vienna
Native name: Wien
```

Q1.4 WHO.IS?

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

```
Florian DUFOUR
REDACTED FOR PRIVACY
```

SOCIETE EDITRICE DU MONDE

Q1.5 CURL

4. Ten first lines:

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

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

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

1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/
MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:dbpprop="http://dbpedia.org/property/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    version="XHTML+RDFa 1.0"
    xml:lang="en"
>
```

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

2.

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

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

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

```
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
       xml:base="http://ns.inria.fr/fabien.gandon">

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

```
3.
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/
MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:dbpprop="http://dbpedia.org/property/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    version="XHTML+RDFa 1.0"
    xml:lang="en"
>
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:dbo="http://dbpedia.org/ontology/"
      xmlns:dbp="http://dbpedia.org/property/"
      xmlns:dct="http://purl.org/dc/terms/"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:skos="http://www.w3.org/2004/02/skos/core#"
```

```
4.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html version="XHTML+RDFa 1.0" xmlns="http://www.w3.org/1999/xhtml"
xmlns:madsrdf="http://www.loc.gov/mads/rdf/v1#" xmlns:ri="http://id.loc.gov/
ontologies/RecordInfo#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:skos="http://
www.w3.org/2004/02/skos/core#" xmlns:skosxl="http://www.w3.org/2008/05/skos-
xl#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:cs="http://www.w3.org/
2003/06/sw-vocab-status/ns#" xmlns:dcterms="http://purl.org/dc/terms/">
  <head>
    <title>Hugo, Victor, 1802-1885 - LC Linked Data Service: Authorities and
Vocabularies | Library of Congress</title>
    <meta name="description" content=" The Linked Data Service provides access
to commonly found standards and vocabularies promulgated by the Library of
Congress. This includes data values and the controlled vocabularies that house
them.   Datasets available include LCSH, BIBFRAME, LC Name Authorities, LC
Classification, MARC codes, PREMIS vocabularies, ISO language codes, and
more."/>
    <link rel="schema.DC" href="http://purl.org/dc/elements/1.1/"/>
    <link rel="dc.relation.isPartOf" href="//www.loc.gov/" title="Library of
Congress"/>
    <meta name="dc.title" content=" LC Linked Data Service: Authorities and
Vocabularies (Library of Congress)"/>
    <meta name="dc.contributor" content="The Library of Congress”/>
```

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <madsrdf:PersonalName rdf:about="http://id.loc.gov/authorities/names/
n79091479" xmlns:madsrdf="http://www.loc.gov/mads/rdf/v1#">
    <rdf:type rdf:resource="http://www.loc.gov/mads/rdf/v1#Authority"/>
    <madsrdf:authoritativeLabel xml:lang="en">Hugo, Victor, 1802-1885</
madsrdf:authoritativeLabel>
    <madsrdf:elementList rdf:parseType="Collection">
      <madsrdf:FullNameElement>
```

```
      <madsrdf:elementValue xml:lang="en">Hugo, Victor,</madsrdf:elementValue>
       </madsrdf:FullNameElement>
       <madsrdf:DateNameElement>
      <madsrdf:elementValue xml:lang="en">1802-1885</madsrdf:elementValue>
```

```
5.
<!DOCTYPE html SYSTEM "about:legacy-compat">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en"
xml:lang="en"><head><title>MCAM - Cell surface glycoprotein MUC18 precursor -
Homo sapiens (Human) - MCAM gene &amp;amp; protein</title><meta content="IE=edge"
http-equiv="X-UA-Compatible"/><meta content="text/html; charset=UTF-8" http-
equiv="Content-Type"/><meta content="width=device-width, initial-scale=1"
name="viewport"/><link href="/" rel="home"/><link href="https://
creativecommons.org/licenses/by/4.0/" rel="license"/><link type="image/
vnd.microsoft.icon" href="/favicon.ico" rel="shortcut icon"/><link href="/
uniprot.min.css2019_08" type="text/css" rel="stylesheet"/><script type="text/
javascript">
                var BASE = '/';
                var ua = window.navigator.userAgent;
                var directory = (~ua.indexOf('MSIE ') ||
~ua.indexOf('Trident/')) === 0 ? "non-ie" : "ie";
           </script><script src="/scripts/frontier/d3/d3.v3.min.js" type="text/
javascript"></script><script src="/js-compr.js2019_08" type="text/
javascript"></script><script type="text/javascript">
                    uniprot.namespace = 'uniprot';
                    uniprot.releasedate = '2019_08';
                </script><script type="text/javascript">
                ;
```

```
<?xml version='1.0' encoding='UTF-8'?>
<rdf:RDF xml:base="http://purl.uniprot.org/uniprot/" xmlns="http://
purl.uniprot.org/core/" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://
www.w3.org/2000/01/rdf-schema#" xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#" xmlns:bibo="http://purl.org/
ontology/bibo/" xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:void="http://
rdfs.org/ns/void#" xmlns:sd="http://www.w3.org/ns/sparql-service-description#"
xmlns:faldo="http://biohackathon.org/resource/faldo#">
<owl:Ontology rdf:about="http://purl.uniprot.org/uniprot/">
<owl:imports rdf:resource="http://purl.uniprot.org/core/"/>
</owl:Ontology>
<rdf:Description rdf:about="http://purl.uniprot.org/uniprot/P43121">
<rdf:type rdf:resource="http://purl.uniprot.org/core/Protein"/>
<reviewed rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</
reviewed>
<created rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1995-11-01</
created>
<modified rdf:datatype="“http://www.w3.org/2001/XMLSchema#date">2019-09-18</
modified>
```

```
6. Xavier Dolan, type: json
```

```
7. Distributed Artificial Intelligence for Distributed Corporate Knowledge
Management, type: turtle/n3
```

Q1.6 Recall five best practices of linked open data

- On the web with open license
- Machine-readable data (structured data)
- Non-proprietary format (ex:csv instead of Excel)
- RDF standards (URIs)
- Linked RDF

Q1.7 Spotlight demo

Reproduce the demo:

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

An ocean (from Ancient Greek Ὠκεανός, transc. Okeanós[1]) is a body of water that composes much of a planet's hydrosphere.[2] On Earth, an ocean is one of the major conventional divisions of the World Ocean. These are, in descending order by area, the Pacific, Atlantic, Indian, Southern (Antarctic), and Arctic Oceans.[3][4] The word "ocean" is often used interchangeably with "sea" in American English. Strictly speaking, a sea is a body of water (generally a division of the world ocean) partly or fully enclosed by land,[5] though "the sea" refers also to the oceans.

Saline water covers approximately 361,000,000 km2 (139,000,000 sq mi) and is customarily divided into several principal oceans and smaller seas, with the ocean covering approximately 71% of Earth's surface and 90% of the Earth's biosphere.[6] The ocean contains 97% of Earth's water, and oceanographers have stated that less than 5% of the World Ocean has been explored.[6] The total volume is approximately 1.35 billion cubic kilometers (320 million cu mi) with an average depth of nearly 3,700 meters (12,100 ft).[7][8][9]

An ocean (from Ancient Greek Ὠκεανός, transc. Okeanós[1]) is a body of water that composes much of a planet's hydrosphere.[2] On Earth, an ocean is one of the major conventional divisions of the World Ocean. These are, in descending order by area, the Pacific, Atlantic, Indian, Southern (Antarctic), and Arctic Oceans.[3][4] The word "ocean" is often used interchangeably with "sea" in American English. Strictly speaking, a sea is a body of water (generally a division of the world ocean) partly or fully enclosed by land,[5] though "the sea" refers also to the oceans.

Saline water covers approximately 361,000,000 km2 (139,000,000 sq mi) and is customarily divided into several principal oceans and smaller seas, with the ocean covering approximately 71% of Earth's surface and 90% of the Earth's biosphere.[6] The ocean contains 97% of Earth's water, and oceanographers have stated that less than 5% of the World Ocean has been explored.[6] The total volume is approximately 1.35 billion cubic kilometers (320 million cu mi) with an average depth of nearly 3,700 meters (12,100 ft).[7][8][9]

Day 02: questions from the course on RDF.

Q2.0 What is the mathematical structure built by the RDF triples?
(give the type of structure and its definition/explanation)

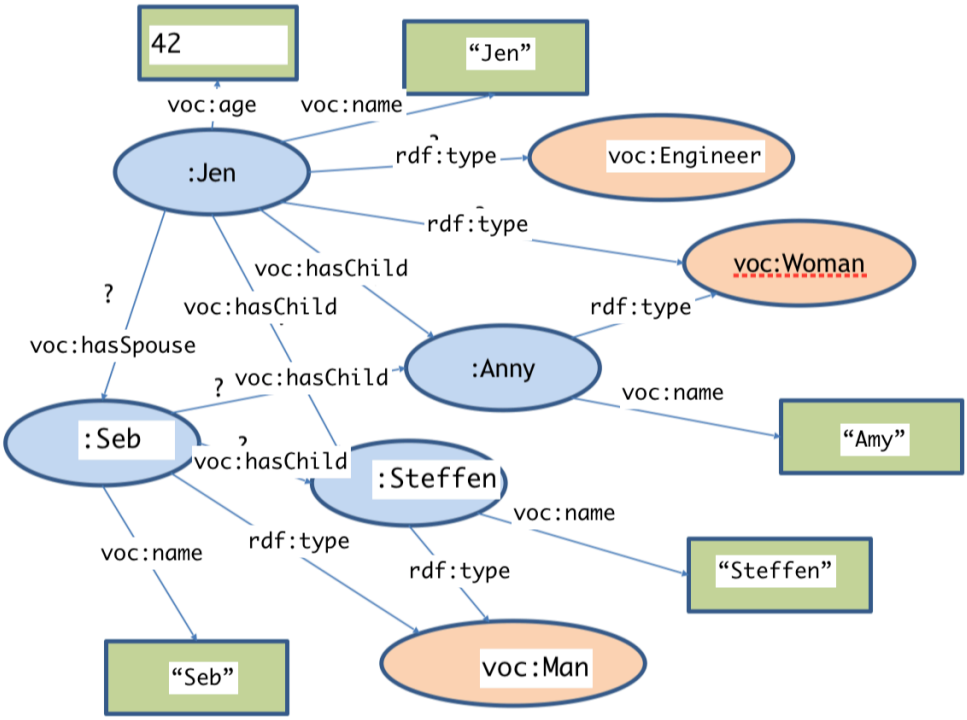
RDF is a model for directed labeled multigraph:
- directed: edges have a direction
- labeled: edges and nodes are labeled
- multigraph: there can be several edges/arcs between nodes/vertices.

Q2.1 Fill the blanks

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

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

For each person we also explicitly specify the name



Q2.2 Fill the blanks (RDF/XML)

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

Q2.3 Fill the blanks (N3/Turtle)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://www.unice.fr/voc#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://www.unice.fr/data#Jen> a voc:Engineer , voc:Woman ;
    voc:age "42"^^xsd:string ;
    voc:hasChild <http://www.unice.fr/data#Anny>, <http://www.unice.fr/
data#Steffen>;
    voc:hasSpouse <http://www.unice.fr/data#Seb> ;
    voc:name "Jen" .
<http://www.unice.fr/data#Seb> a voc:Man ;
    voc:hasChild <http://www.unice.fr/data#Anny>,
        <http://www.unice.fr/data#Steffen> ;
    voc:name "Seb" .
<http://www.unice.fr/data#Anny> a voc:Woman ;
    voc:name "Anny" .
<http://www.unice.fr/data#Steffen> a voc:Man ;
     voc:name "Steffen" .
```

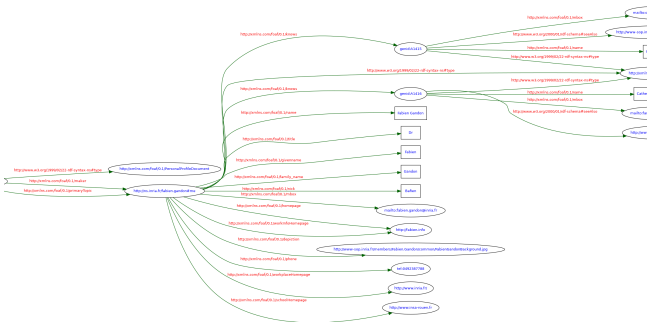
Q2.4 Visit me please

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

1. Get the RDF data from: http://ns.inria.fr/fabien.gandon#me
2. What is the syntax used?
3. Validate it and see the graph:
   http://www.w3.org/RDF/Validator/

4. Translate into Turtle/N3:
   http://rdf-translator.appspot.com/
   http://www.easyrdf.org/converter
5. Visualize it also with:
   http://cltl.nl/visualrdf/
   http://www.easyrdf.org/converter (PNG, SVG)
6. Adapt to your data and do it again

1. OK
2. RDF XML
3. OK, partial screenshot:

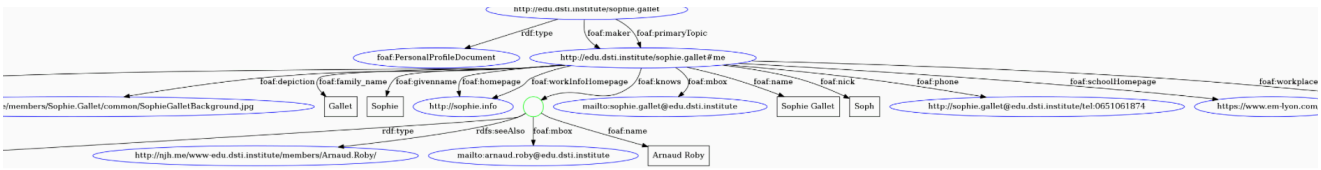


4. OK, first few lines:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<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:depiction <http://www-sop.inria.fr/members/Fabien.Gandon/common/
FabienGandonBackground.jpg> ;
    foaf:family_name "Gandon" ;
    foaf:givenname "Fabien" ;
    foaf:homepage <http://fabien.info> ;
```

5. OK, partial screenshot:

6. OK, see partial screenshot:



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

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

"http://example.org/doc.html" is a report with a Music theme and a Danse theme. It has 73 pages.

Q2.6 Visit to Victor Hugo

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

1. OK
2. OK
3. RDF XML
4. OK
5. Problem with the translated names of Victor Hugo (claimed it is in English when not the case)

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

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

TriG.
http://inria.fr/rr/doc.html has the subject "http://data.bnf.fr/ark:/12148/cb121105993" (algebra according to BNF) and belongs to the graph "http://inria.fr/topics/algebra"

Q2.8 Visit Leukocyte surface antigen CD53

1. See HTML data from:
   http://www.uniprot.org/uniprot/Q61451
2. Get RDF data from:
   http://www.uniprot.org/uniprot/Q61451.rdf
3. What is the syntax?

4. Translate into Turtle/N3:
   http://rdf-translator.appspot.com/
5. Any remark about the structure of the data?

1. OK
2. OK
3. RDF XML
4. OK
5. They use reification (deprecated). This can be seen searching for rdf:Statement

---

Day 03: questions from the course on SPARQL.

Q3.1 Test SPARQL online

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

Answers to the query:

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

	x
1	<http://www.inria.fr/2015/humans-instances#John>
2	<http://www.inria.fr/2015/humans-instances#Sophie>
3	<http://www.inria.fr/2015/humans-instances#Mark>
4	<http://www.inria.fr/2015/humans-instances#Eve>
5	<http://www.inria.fr/2015/humans-instances#David>
6	<http://www.inria.fr/2015/humans-instances#Laura>
7	<http://www.inria.fr/2015/humans-instances#William>
8	<http://www.inria.fr/2015/humans-instances#Karl>

Q3.2 Test SPARQL online

Connect to

http://dbpedia.org/snorql/ or

http://fr.dbpedia.org/sparql or ...

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

Answers to the query:

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

x	p	v
:Paris	rdf:type	owl:Thing
:Paris	rdf:type	dbpedia:ontology/Place
:Paris	rdf:type	dbpedia:ontology/Location
:Paris	rdf:type	<http://www.wikidata.org/entity/Q486972>
:Paris	rdf:type	dbpedia:ontology/PopulatedPlace
:Paris	rdf:type	dbpedia:ontology/Settlement
:Paris	rdf:type	<http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing>
:Paris	rdf:type	<http://schema.org/Place>
:Paris	rdf:type	<http://umbel.org/umbel/rc/Location_Underspecified>
:Paris	rdf:type	<http://umbel.org/umbel/rc/PopulatedPlace>

Q3.3 Test SPARQL online
Connect to:
https://query.wikidata.org/
What does this query retrieve?

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

Discover wd:Q30 using the namespace attached to wd:
PREFIX wd: <http://www.wikidata.org/entity/>
Discover p:P6 using the namespace attached to p:
PREFIX p: <http://www.wikidata.org/prop/>
Find q-name of the property “given name” https://www.wikidata.org/wiki/Wikidata:List_of_properties

```
Retrieve 44 of the American presidents (there should be 45..).

https://www.wikidata.org/wiki/Q30: United States of America
https://www.wikidata.org/wiki/Property:P6: Head of government

The q-name of “given name” is P735
```

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

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

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

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

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

1.Find the age of resources whose name is ‘Fabien’

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

3.Find property values of resources whose name is ‘Fabien’ and whose age is less than 50

4.Find other names of resources whose name is ‘Fabien’

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

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

```
1.
SELECT ?age WHERE {
      ?x c:name ‘Fabien’ ;
          c:age ?age .
}
2.
SELECT ?name WHERE {
      ?x c:name ?name ;
          c:age ?age .
      FILTER(?age <50)
}
3.
SELECT ?p ?v WHERE {
      ?x c:name ‘Fabien’ ;
          c:age ?age ;
          ?p ?v .
      FILTER(?age <50)
}
4.
SELECT ?name WHERE {
      ?x c:name ‘Fabien’ ,
                ?name .
      FILTER(?name != ‘Fabien’)
}
5.
SELECT * WHERE {
      ?x ?p ?v ;
          ?q ?v .
      FILTER(p != ?q)
}
6.
SELECT * WHERE {
      ?x ?p ?y ,
              ?z .
```

```
    FILTER(?y != ?z)
}
```

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

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

Yes it could, as there are no common variables between the two patterns.

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

```
SELECT ?x WHERE {
    ?x c:memberOf ?org .
    filter (! exists{
        ?x c:author ?doc
    })
}
```

Q3.9 what is retrieving this query ?

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

It will return authors and the number of documents produced respectively, with the most prolific authors first.

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

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

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

Q3.11 what is this query retrieving?

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

It retrieves all the people who know someone else, along with the name of that friend if name is known, and "John Doe" otherwise.

Q3.12 what is this query retrieving?

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

It retrieves people along with the average age of the people they know.

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

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

```
1.
SELECT * WHERE {
     {?x c:study c:Informatics}
     UNION
     {?x c:study c:Informatics}
}
2.
SELECT * WHERE {
     {?x c:study c:Informatics}
     UNION
     {?x c:study c:Informatics}
     OPTIONAL {?x c:name ?name}
}
3.
SELECT * WHERE {
     {?x c:study c:Informatics}
     UNION
     {?x c:study c:Informatics}
     OPTIONAL {GRAPH ?g
               {?x c:name ?name}
```

```
                }
}
```

Q3.14 On which graph(s) is calculated ?x ?p ?y
On which graph(s) is calculated graph ?g { ?y ?q ?z }

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

g1 as it is not in the graph {} and thus using FROM
g2 as it is in the graph {} and thus using FROM NAMED

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

```
delete { ?x foaf:name ?n }
insert { ?x rdfs:label ?n }
where { ?x foaf:name ?n }
```

Q3.16 what is this query performing?

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

It convert the age property values that are strings into integers.

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

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

```
Construct{} Where{}
```

Day 04: questions from the course on RDFS.

Q4.1 Choose among the following assertions one or more you consider to be true:

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

- an ontology may allow inferences on data that uses it
- conceptual graphs can represent an ontology
- a shared ontology promotes interoperability
- description logics can represent an ontology (one of the biggest family to represent ontology)

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

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

- describe classes of resources
- describe types of properties of resources
- document definitions in natural language

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

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

Defined:

- Phone is a subclass of device,
- Computer is a subclass of device
- Smartphone is a subclass of computer and phone

Derived:

- Device, Phone, Computer and Smartphone are resources
- Smartphone is a subclass of device

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

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

Defined:

- employeeOf, hasControlOver are sub properties of proRelationWith
- isShareholderOf is a sub property of hasControlOver
- isCEOof is a sub property of employeeOf and hasControlOver

Derived:

- isCEOof, isShareholderOf are sub properties of proRelationWith
- All the properties are resources

Q4.5. What can be said about the types of the resources that will be linked by the properties defined below?

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

- Resources that are attributes of piloteOf will be of type Object and Plane.
- Resources that are attributes of driverOf will be of type Object and Car.
- Resources that are the subject of piloteOf will be of type Human Adult.
- Resources that are the subject of driverOf will be of type Human.

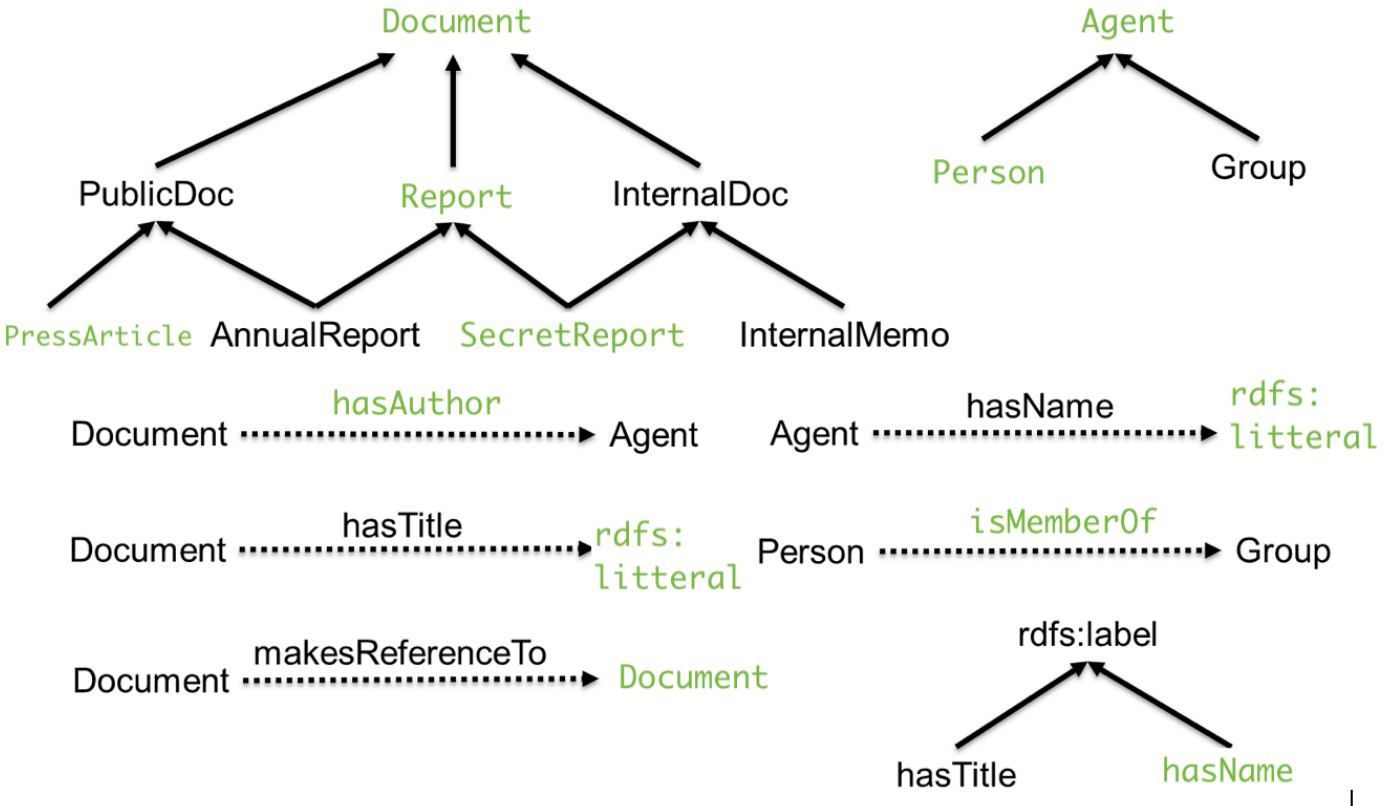
Q4.6. What could we add to this schema (several answers are possible)?

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
@base  <http://inria.fr/2005/humans.rdfs>
<p1>  a rdf:Property ;  rdfs:label  "age"@fr .
<c1>  a rdfs:Class; rdfs:comment "un être humain"@fr .
```

- <p1> rdfs:label "prénom"@fr .
- <c1> rdfs:comment "a human being"@fr .
- <c1> rdfs:label "personne"@fr .
- <p1> rdfs:label "age"@en .
- <c1> rdfs:label "woman"@en .
- <c1> rdfs:label "persona"@es .

- `<c1> rdfs:label "personne"@fr .`
- `<p1> rdfs:label "age"@en .`
- `<c1> rdfs:label "persona"@es .`

Q4.7. (a) Fill the blanks with: `Document, PublicDoc, PressArticle, Report, AnnualReport, InternalDoc, SecretReport, InternalMemo, Agent, Person, Group, hasTitle, hasAuthor, makesReferenceTo, hasName, isMemberOf` + rdf / rdfs primitives.
(b) Write it in RDFS and validate the RDF.



Day 04: questions from the course on OWL.

Q5.1 What can we deduce?

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

Jane is a Woman.

John is Human and a Male.

Q5.2 What are we defining and inferring?

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

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

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

Defining: GrandFather/Father is a subclass of Parent and Man.

Inferring: Jack is a Man and a Parent.

Q5.3 What can we deduce?

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

- Jane hasParent Jim. Jane hasAncestor Jim and James
- John hasSpouse Jane
- Jim hasAncestor James. James hasChild Jim.

Q5.4 What can we deduce?

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

JimmyPage are the same, and they are a Person.

JimmyHendrix is different from JimmyPage and thus from JamesPatrickPage.

Q5.5 What are we defining and inferring?

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

The UnluckyPerson class is equivalent to the anonymous class that is the intersection of a Person Class and of the anonymous Class that is the complement of Lucky. More clearly, an unlucky person is two things: a person and the complement of lucky.

Q5.6 What can we deduce?

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

Tom’s parents are all humans, so James and Jane are humans.

Q5.7 What are we defining and inferring?

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

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

The list (ex:a ex:b ex:c) is a PersonList. ex:a is a Person, the rest is a PersonList. Thus ex:b is a Person and ex:c is a PersonList. Thus ex:c is a Person. In brief, all the elements in the list are Persons.

Q5.8 What are we defining and inferring?

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

A human can have only 1 biological father and only 1 biological mother.

The system will infer that James and John are the same person. If you specify they're sameAs, then the system will find an error.

Q5.9 What are we defining and inferring?

```
@prefix ex: <http://example.org/>
ex:Wealthy a owl:Class ;
     owl:equivalentClass  [
       a owl:Class ; owl:intersectionOf  (
         [ a owl:Restriction ;
           owl:onProperty ex:hasChild ;
           owl:allValuesFrom ex:Wealthy
         ] ,
         [ a owl:Restriction ;
           owl:onProperty ex:hasChild ;
           owl:someValuesFrom ex:Wealthy
         ]
     )] .
ex:Tom a ex:Wealthy ; ex:hasChild ex:Tim .
```

To be wealthy, all of the resource's children need to be wealthy, and there must be at least one value of a wealthy child. Thus Tim is wealthy.

Day 05: questions from the course on Vocabularies.

Q6.1 What do you think of the annotation?

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

There's a problem: there are two preferred labels in English.

Q6.2 practice:

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

1.
http://www.w3.org/2004/02/skos/core#
2. OK
3.

- Inverse of broadMatch
- Sub property of mappingRelation
- Sub property of narrower

Q6.3 practice:

1. Open the source file of Dublin Core Terms:
   http://dublincore.org/2012/06/14/dcterms.rdf
   Look at the definition of the class `FileFormat` and find the class it inherits from.
2. Choose your preferred book on Amazon, Fnac, etc. and describe it in an RDF annotation using as many DC primitives as necessary .
3. Add the most restrictive CC license to your preferred book ; is this license appropriate?

1. It inherits from MediaType

2 & 3.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX cc: <http://creativecommons.org/ns#>
PREFIX xhtml:license <http://creativecommons.org/ns#>
```

```
<https://www.amazon.com/Giver-Quartet-Lois-Lowry>


dc:creator <http://ns.greatauthor.com/lois.lowry#me> ;

dc:title “The Giver"@en ; dc:language "en" ;

dc:subject <#ScienceFiction> ;

dc:date "1993-01-01" ;

dc:publisher <http://www.hmhco.com> ;

dc:format "text/html" ;

dc:type dcterms:Text ;

xhtml:license <xhtml:license rdf:resource="http://creativecommons.org/licenses/
by-nc-nd/4.0/> .


CC BY-NC-ND: content cannot be changed in any way or used for commercial
purposes.
```

Q6.4 practice:

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

```
2. Types: ObjectProperty, InverseFunctionalProperty,

3. Inherits from page

4. The signature of weblog is domain:Agent; range:Document. The signature of
page is domain:Thing; range:Document

InverseFunctionalProperty: if a blog has an author, and the same blog has a
second author, it will be inferred that both authors are the same.
```

Q6.5 practice:

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

```
@prefix admin: <http://webns.net/mvcb/> .

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rel: <http://purl.org/vocab/relationship#> .

<> a foaf:PersonalProfileDocument ;
    admin:errorReportsTo <mailto:leigh@ldodds.com> ;
    admin:generatorAgent <http://www.ldodds.com/foaf/foaf-a-matic> ;
    foaf:maker <#me> ;
    foaf:primaryTopic <#me> .

<#me> a foaf:Person ;
    foaf:family_name "Gallet" ;
    foaf:givenname "Sophie" ;
    foaf:knows [ a foaf:Person ;
            foaf:mbox <mailto:vidhya.kannan@edu.dsti.institute> ;
            foaf:name "Vidhya" ] ;
    foaf:mbox <mailto:sophiemary.gallet@gmail.com> ;
    foaf:name "Sophie Gallet" ;
    foaf:nick "Fof" ;
    foaf:schoolHomepage <https://www.datasciencetech.institute> ;
    foaf:title "Ms" ;
rel:SiblingOf "Nicolas";
rel:EnemyOf "Donald Trump";
rel:lifePartnerOf "Brendan Loudermilk";
rel:neighborOf "Oliver";
rel:influencedBy "Alexandra Ocasio Cortez";
rel:apprenticeTo "Fabien Gandon".
```

Q6.6 What does this mean?

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

BioRDF2DBL is linked to two targets BioRDF and DBLP. The predicate used to link to the two databases is exact match. There are close to 9000 triples in the database.

Q6.7 practice:

1. Connect to the Void Store SPARQL endpoint: http://void.rkbexplorer.com/sparql/
2. What is the meaning of the default SPARQL query in the interface, run it and look at the results.
3. Write a SPARQL query to find the dataset that has for label "DBpedia-fr" and all its properties.

It retrieves the distinct SPARQL endpoints for data sets that have a sparql endpoint.

```
PREFIX rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:      <http://www.w3.org/2002/07/owl#>
PREFIX xsd:      <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf:     <http://xmlns.com/foaf/0.1/>
PREFIX dcterms:  <http://purl.org/dc/terms/>
PREFIX scovo:    <http://purl.org/NET/scovo#>
PREFIX void:     <http://rdfs.org/ns/void#>
PREFIX akt:      <http://www.aktors.org/ontology/portal#>
```

```
SELECT * WHERE { ?ds a void:Dataset ; rdfs:label  'DBpedia-fr' ; ?p ?v.}
```

Q6.8 What does this mean?

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

plot used stats1998. The bar-chart was generated by the plot. Stats1998 is a distribution in csv format and of mediaType text/csv.

Q6.9 What does this mean?

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

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

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

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

Db-employ is a DCAT distribution that can be download using the URL provided, has for total “SQL Dump of the employees” was issued in 2015 in Sophia (found following the spatial URL), has for publisher Inria.fr, is of type/format SQL, is in French and it has a size.

R2RTransform12 uses 3 resources: db-employ, R2R-employ-mapping and what can be found at given URL.

FoaFDump is a rdf dataset with about 13K triples. Its title is “RDF Dump of the employees”. It was generated by R2RTransform12, in 2015 and id derived from db-employ.

(Not exhaustive description, but the most important is there)

Q6.10 practice:

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

```
mo:MusicArtist
version: 2.1.5
Reused: af and theatre
54 classes and 153 properties.
RDF RDFS and OWL
```

Day 05: questions from the course on other data formats.

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

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

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

```
Two triples:
https://www.ietf.org/rfc/rfc2616.txt dc:title ‘Hypertext Transfer Protocol --
HTTP/1.1’
https://www.ietf.org/rfc/rfc2396.txt dc:title ‘Uniform Resource Identifiers
(URI): Generic Syntax’
```

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

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

```
cathy rdf:type Person
cathy rdf:name 'Catherine Faron'
cathy rdf:mbox 'faron@i3s.unice.fr'
cathy rdf:knows 'Fabien Gandon'
```

Q7.3 practice:

1. Look at the Web Page
   https://www.w3.org/TR/xhtml-rdfa-scenarios/scenario-2.html
2. Call the translator on this Web page to get Turtle:
   http://rdf-translator.appspot.com/
3. What does the extracted triple say?
4. Do the same with:
   http://schema.org/docs/schema_org_rdfa.html
   What kind of data is represented in that page?
5. Again, what are the different subjects described in RDFa in this page:
   http://iricelino.org/rdfa/sample-annotated-page.html

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

Scenario-2 has for creator 'Paul' (English language)

4. An ontology

5. Spinoza, Einstein, Schopenhauer among others.
```

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

```
<span vocab="http://schema.org/" typeof="TechArticle">
   <a property="url" href="http://www.w3.org/TR/rdfa-primer/">
      <span property="name">RDFa 1.1 Primer</span></a>
       <span property="creator">Sophie</span>.
```

```
</span>
```

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

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

With RDFa Distiller and Parser

```
@prefix fb: <http://www.facebook.com/2008/fbml> .
@prefix ns1: <http://www.facebook.com/2008/> .
@prefix og: <http://ogp.me/ns#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xlink: <http://www.w3.org/1999/xlink> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<https://www.imdb.com/title/tt0137523/?ref_=nv_sr_1?ref_=nv_sr_1>
og:description "Directed by David Fincher.  With Brad Pitt, Edward Norton, Meat
Loaf, Zach Grenier. An insomniac office worker and a devil-may-care soapmaker
form an underground fight club that evolves into something much, much more." ;
    og:image "https://m.media-amazon.com/images/M/
MV5BMmEzNTkxYjQtZTc0MC00YTVjLTg5ZTEtZWMwOWVlYzY0NWIwXkEyXkFqcGdeQXVyNzkwMjQ5NzM
@._V1_UY1200_CR85,0,630,1200_AL_.jpg" ;
    og:site_name "IMDb" ;
    og:title "Fight Club (1999) - IMDb" ;
    og:type "video.movie" ;
    og:url "http://www.imdb.com/title/tt0137523/" ;
    ns1:fbmlapp_id "115109575169727" .
```

With the translator

```
@prefix fb: <http://www.facebook.com/2008/fbml> .
@prefix ns1: <http://www.facebook.com/2008/> .
@prefix og: <http://ogp.me/ns#> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xlink: <http://www.w3.org/1999/xlink> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<https://www.imdb.com/title/tt0137523/?ref_=nv_sr_1?ref_=nv_sr_1>
og:description "Directed by David Fincher.  With Brad Pitt, Edward Norton, Meat
Loaf, Zach Grenier. An insomniac office worker and a devil-may-care soapmaker
form an underground fight club that evolves into something much, much more." ;
    og:image "https://m.media-amazon.com/images/M/
MV5BMmEzNTkxYjQtZTc0MC00YTVjLTg5ZTEtZWMwOWVlYzY0NWIwXkEyXkFqcGdeQXVyNzkwMjQ5NzM
@._V1_UY1200_CR85,0,630,1200_AL_.jpg" ;
    og:site_name "IMDb" ;
    og:title "Fight Club (1999) - IMDb" ;
    og:type "video.movie" ;
    og:url "http://www.imdb.com/title/tt0137523/" ;
    ns1:fbmlapp_id "115109575169727" .
```

Q7.6 Test JSON-LD online

1. Transform your FOAF profile in JSON-LD with the translator: http://rdf-translator.appspot.com/
2. Use the following online tool to generate different variations of JSON-LD of your profile (expanded, collapsed, flattened, etc.) http://json-ld.org/playground/

Compacted:

```
<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://purl.org/vocab/relationship#EnemyOf> "Donald Trump" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://purl.org/vocab/relationship#SiblingOf> "Nicolas" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://purl.org/vocab/relationship#apprenticeTo> "Fabien Gandon" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://purl.org/vocab/relationship#influencedBy> "Alexandra Ocasio Cortez" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://purl.org/vocab/relationship#lifePartnerOf> "Brendan Loudermilk" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://purl.org/vocab/relationship#neighborOf> "Oliver" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/
Person> .
```

```
<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/family_name> "Gallet" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/givenname> "Sophie" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/knows> _:b0 .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/mbox> <mailto:sophiemary.gallet@gmail.com> .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/name> "Sophie Gallet" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/nick> "Fof" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/schoolHomepage> <https://
www.datasciencetech.institute> .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/#me>
<http://xmlns.com/foaf/0.1/title> "Ms" .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/> <http://
webns.net/mvcb/errorReportsTo> <mailto:leigh@ldodds.com> .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/> <http://
webns.net/mvcb/generatorAgent> <http://www.ldodds.com/foaf/foaf-a-matic> .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/> <http://
www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/
PersonalProfileDocument> .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/> <http://
xmlns.com/foaf/0.1/maker> <file:///base/data/home/apps/s%7Erdf-translator/
2.408516547054015808/#me> .

<file:///base/data/home/apps/s%7Erdf-translator/2.408516547054015808/> <http://
xmlns.com/foaf/0.1/primaryTopic> <file:///base/data/home/apps/s%7Erdf-
translator/2.408516547054015808/#me> .

_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/
0.1/Person> .

_:b0 <http://xmlns.com/foaf/0.1/mbox> <mailto:vidhya.kannan@edu.dsti.institute>
.

_:b0 <http://xmlns.com/foaf/0.1/name> "Vidhya" .
```

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

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

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

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

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

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

and we receive the following answer:

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

Which ones of the following statements are true?

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

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

PRACTICAL SESSIONS

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: http://wimmics.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 voc: <http://www.dsti.institute/voc#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://www.dsti.institute/data#Jen> a voc:Engineer , voc:Woman ;
    voc:age "42"^^xsd:string ;
    voc:hasChild <http://www.dsti.institute/data#Anny>, <http://
www.dsti.institute/data#Steffen>;
    voc:hasSpouse <http://www.dsti.institute/data#Seb> ;
    voc:hasFather <http://www.dsti.institute/data#Thomas> ;
```

```
    voc:shoeSize "36"^^xsd:string ;
    voc:trouserSize "38"^^xsd:string ;
    voc:hasColleague <http://www.dsti.institute/data#Catherine> ;
    voc:hasColleague <http://www.dsti.institute/data#Fabien> ;
    voc:name "Jen" .
<http://www.dsti.institute/data#Seb> a voc:Man ;
    voc:hasChild <http://www.dsti.institute/data#Anny>,
        <http://www.dsti.institute/data#Steffen> ;
    voc:name "Seb" .
<http://www.dsti.institute/data#Anny> a voc:Woman ;
    voc:name "Anny" .
<http://www.dsti.institute/data#Steffen> a voc:Man ;
     voc:name "Steffen" .
<http://www.dsti.institute/data#Thomas> a voc:Man ;
     voc:name "Thomas" .
```

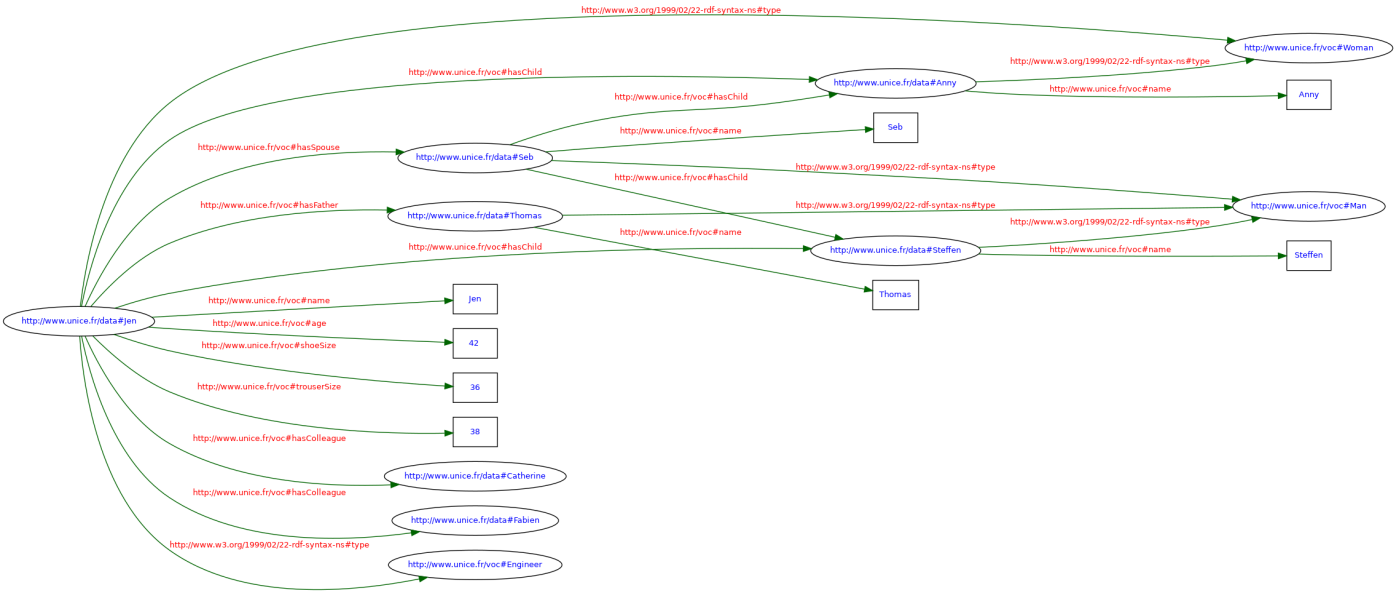
Code of validated RDF in XML syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [<!ENTITY vocab "http://www.unice.fr/voc"> <!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#"> ]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:voc="&vocab;#" xml:base="http://www.unice.fr/data">
 <voc:Woman rdf:about="#Jen">
    <voc:name>Jen</voc:name>
    <voc:age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42 </
voc:age>
    <voc:shoeSize rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">36</
voc:shoeSize>
    <voc:trouserSize rdf:datatype="http://www.w3.org/2001/
XMLSchema#integer">38</voc:trouserSize>
    <voc:hasSpouse rdf:resource="#Seb"></voc:hasSpouse>
    <voc:hasColleague rdf:resource="#Catherine"></voc:hasColleague>
    <voc:hasColleague rdf:resource="#Fabien"></voc:hasColleague>
    <voc:hasFather rdf:resource="#Thomas"></voc:hasFather>
    <voc:hasChild rdf:resource="#Steffen"></voc:hasChild>
    <voc:hasChild>
      <rdf:Description rdf:about="#Anny">
        <voc:name>Anny</voc:name>
        <rdf:type rdf:resource="&vocab;#Woman"></rdf:type>
```

```
      </rdf:Description>
    </voc:hasChild>
    <rdf:type rdf:resource="&vocab;#Engineer"></rdf:type>
 </voc:Woman>
 <voc:Man rdf:about="#Seb">
    <voc:name>Seb</voc:name>
    <voc:hasChild rdf:resource="#Steffen"></voc:hasChild>
    <voc:hasChild rdf:resource="#Anny"></voc:hasChild>
  </voc:Man>
  <voc:Man rdf:about="#Steffen">
    <voc:name>Steffen</voc:name>
  </voc:Man>
  <voc:Man rdf:about="#Thomas">
    <voc:name>Thomas</voc:name>
  </voc:Man>
</rdf:RDF>
```

The translated code is much longer than the code I wrote, and doesn't use shortcuts.

The RDF XML graph:



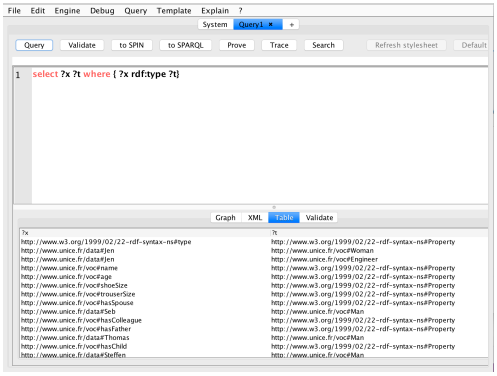
Query your data

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 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<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:depiction <http://www-sop.inria.fr/members/Fabien.Gandon/common/
FabienGandonBackground.jpg> ;
    foaf:family_name "Gandon" ;
    foaf:givenname "Fabien" ;
    foaf:homepage <http://fabien.info> ;
    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:mbox <mailto:fabien.gandon@inria.fr> ;
    foaf:name "Fabien Gandon" ;
    foaf:nick "Bafien" ;
    foaf:phone <http://ns.inria.fr/tel:0492387788> ;
    foaf:schoolHomepage <http://www.insa-rouen.fr> ;
    foaf:title "Dr" ;
    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

```
http://ns.inria.fr/fabien.gandon has for author and primary topic http://
ns.inria.fr/fabien.gandon#me
```

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

```
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?

```
"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?

```
Using rdf:ID
```

3. What is the namespace of the vocabulary used to describe the resources in the dataset and how is it associated with the tags?

```
As there is no prefix before each resource vocabulary, the default one is
invoked: "http://www.inria.fr/2007/09/11/humans.rdfs" (see answer 4.)
```

4. Explain the code `xmlns="&humans;#"`

```
The default namespace is the url between brackets. Here it actually refers to a
shortcut defined at the beginning of the document with the code <!ENTITY humans
"http://www.inria.fr/2007/09/11/humans.rdfs">. Hence the default namespace is
http://www.inria.fr/2007/09/11/humans.rdfs
```

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

In summary, in human language:

John is 37, he's the father of Mark, the friend of Alice, the husband of Jennifer, his parents are Sophie and Harry. Also he has a shoe size of 14, a shirt size of 12, a trouser size of 44.

From the file:

```
<Person rdf:ID="John">
  <name>John</name>
      <shoesize rdf:datatype="&xsd;integer" >14</shoesize>
      <age rdf:datatype="&xsd;integer" >37</age>
</Person>

<Person rdf:ID="Mark">
  <name>Mark</name>
      <shoesize rdf:datatype="&xsd;integer" >8</shoesize>
      <age rdf:datatype="&xsd;integer" >14</age>
      <shirtsize rdf:datatype="&xsd;integer" >9</shirtsize>
      <trouserssize rdf:datatype="&xsd;integer" >36</trouserssize>
      <hasFather rdf:resource="#John"/>
</Person>

<Woman rdf:ID="Alice">
      <hasFriend rdf:resource="#John"/>
      <name>Alice</name>
</Woman>

<Woman rdf:ID="Jennifer">
      <hasSpouse rdf:resource="#John"/>
      <name>Jennifer</name>
</Woman>

<Person rdf:about="#John">
      <shirtsize rdf:datatype="&xsd;integer" >12</shirtsize>
      <trouserssize rdf:datatype="&xsd;integer" >44</trouserssize>
      <hasParent rdf:resource="#Sophie"/>
</Person>
```

```
<Man rdf:ID="Harry">
  <name>Harry</name>
  <hasChild rdf:resource="#John"/>
  <hasSpouse rdf:resource="#Sophie"/>
</Man>
```

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 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@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> ;
```

7. In the turtle version find *everything* about Laura.
   all the information:

In summary, in human language:

Laura is a lecturer, person and researcher. She's married to William, is the mother of Catherine, has a friend Alice.

From the file:

```
<http://www.inria.fr/2007/09/11/humans.rdfs-instances#William> a :Person ;
    :age 42 ;
    :hasSpouse <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura> ;
    :name "William" ;
    :shirtsize 13 ;
    :shoesize 10 ;
    :trouserssize 46 .

<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine> a :Woman ;
    :hasMother <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura> .

<http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura> a :Lecturer,
        :Person,
```

```
        :Researcher ;
    :hasFriend <http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice> ;
    :name "Laura" .
```

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: http://wimmics.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:

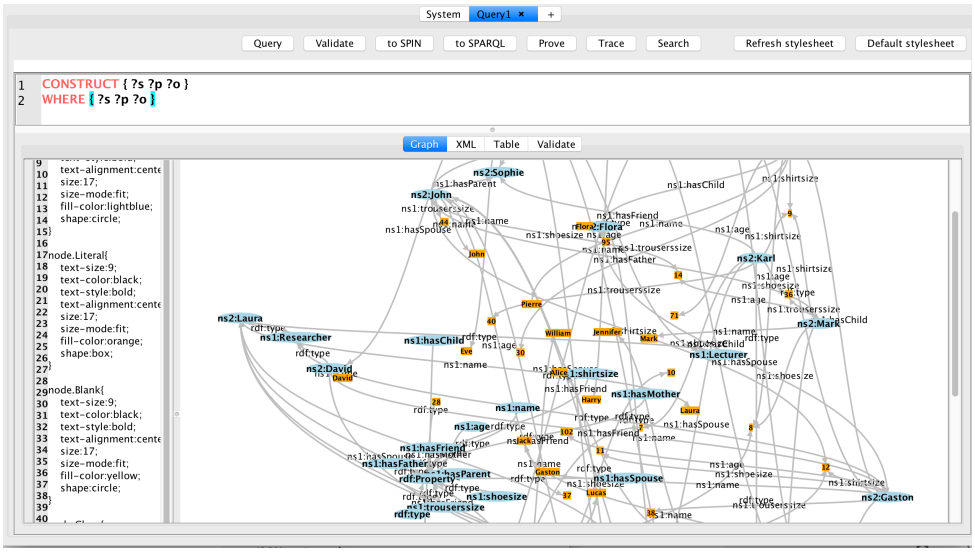
Create a new tab to enter the following query and explain what it does and the results you get. This is a good way to familiarize yourself with the data.

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

Explanation:

```
It displays all the data in the rdf.
```

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.

Select all the resources that have any types, and whose type URI starts with "http://www.inria.fr/2007/09/11/humans.rdfs#". We get in the first column (?x) humans' names (such as Eve, David, …) and in the second column we get their type (Woman, Person, Lecturer, …).

Run this query. How many answers do you get?

I get 21 answers, 2 columns.

Find John and his types in the answers.

John has 1 type: 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 * WHERE {
      {<http://www.inria.fr/2007/09/11/humans.rdfs-instances#John> ?p ?v}
      UNION
      {?x ?q <http://www.inria.fr/2007/09/11/humans.rdfs-instances#John>}
}
```

Results:

37	http://www.inria.fr/2007/09/11/humans.rdfs#age		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie	http://www.inria.fr/2007/09/11/humans.rdfs#hasParent		
John	http://www.inria.fr/2007/09/11/humans.rdfs#name		
12	http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize		
14	http://www.inria.fr/2007/09/11/humans.rdfs#shoesize		
44	http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize		

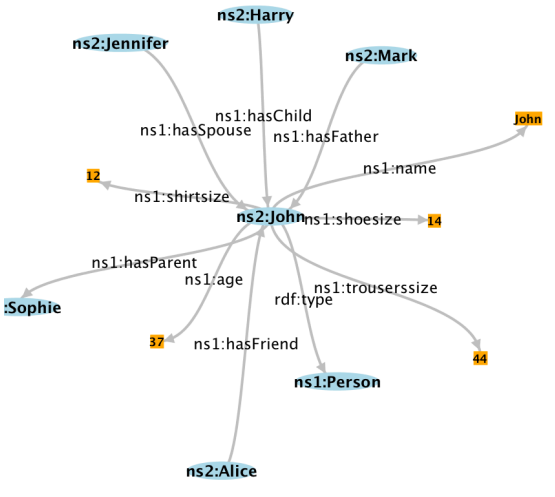
http://www.inria.fr/2007/09/11/humans.rdfs#Person	http://www.w3.org/1999/02/22-rdf-syntax-ns#type		
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs#hasChild
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs#hasFather
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice	http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer	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:



37	http://www.inria.fr/2007/09/11/humans.rdfs#age		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie	http://www.inria.fr/2007/09/11/humans.rdfs#hasParent		
John	http://www.inria.fr/2007/09/11/humans.rdfs#name		

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie	http://www.inria.fr/2007/09/11/humans.rdfs#hasParent		
John	http://www.inria.fr/2007/09/11/humans.rdfs#name		
12	http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize		
14	http://www.inria.fr/2007/09/11/humans.rdfs#shoesize		
44	http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize		
http://www.inria.fr/2007/09/11/humans.rdfs#Person	http://www.w3.org/1999/02/22-rdf-syntax-ns#type		
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs#hasChild
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs#hasFather
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice	http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer	http://www.inria.fr/2007/09/11/humans.rdfs#hasSpouse

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.

```
Select all the resources who have a spouse or are a spouse of someone.
```

Run this query. How many answers do you get?

```
6 answers, 2 columns.
```

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 * where {
      ?x a h:Person;
          h:shoesize ?s.
}
```

Result:

http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	14
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	7
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	8
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	10

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 * where {
      ?x a h:Person
      OPTIONAL {?x h:shoesize ?s }
}
```

Result:

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve	
http://www.inria.fr/2007/09/11/humans.rdfs-instances#David	
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	14
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	7
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	8
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	10
http://www.inria.fr/2007/09/11/humans.rdfs-instances#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:

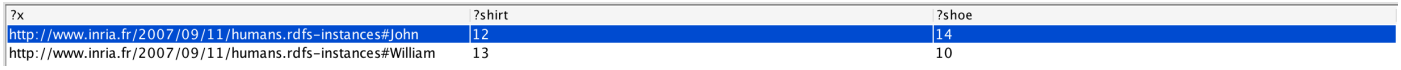
```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
```

```
SELECT * WHERE {
      ?x a h:Person .
      OPTIONAL  {?x h:shirtsize ?shirt }
      OPTIONAL  {?x h:shoesize ?shoe }
      FILTER(?shirt >12 || ?shoe >8)
}
```

Result:

http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	12	14
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	13	10

Screenshot:



?x	?shirt	?shoe
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	12	14
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	13	10

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, with 1 duplicate (Gaston, who has 2 children).
```

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
```

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 .
      MINUS{?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.

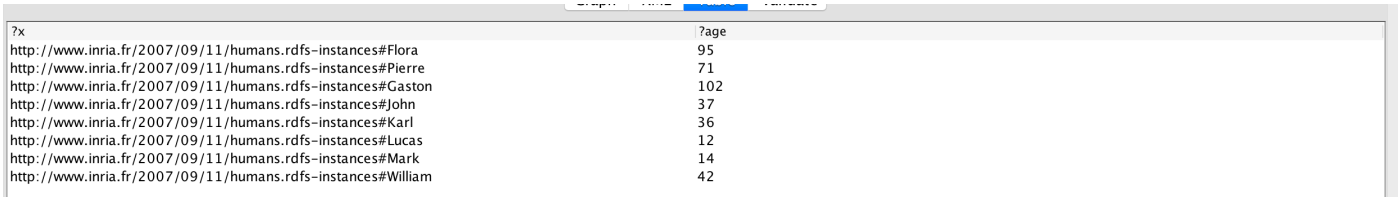
   Query:

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

   Result:

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora	95
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre	71
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	102
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	37
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	36
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	12
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	14
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	42

screenshot:



?x	?age
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora	95
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre	71
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	102
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	37
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	36
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	12
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	14
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	42

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 h:age ?age
      FILTER(?age<18)
}
```

   How many answers do you get?

```
2
```

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

SELECT * ( floor(?age/2) = ?age/2 as ?even)
WHERE {
      ?x a h:Person ; h:age ?age.
}
```

Question 8

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

Query:

```
PREFIX h: <http://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:

```
Select query before/after:

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

Insert query:

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


It works!
```

Question 9

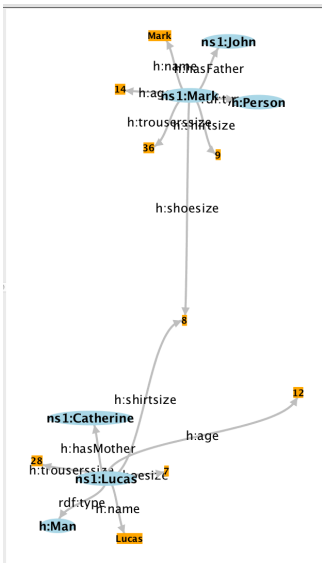
Choose and edit one of the SELECT WHERE queries previously written to transform them into a CONSTRUCT WHERE query (retaining the same WHERE clause) in order to visualize the results as a graph.

Query:

```
I've decided to visualize the information we have on those who are not adults:

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

```
CONSTRUCT {?x ?p ?v} where {
        ?x ?p ?v;
        h:age ?age.
        FILTER(?age <18)
}
```

Result:



Question 10

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

screenshots:

```
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:Sophie a h:Person;
                h:name 'Sophie';
                h:hasFriend 'Vidhya';
                h:hasFather 'Christophe';
                h:shoesize 7;
                h:age 29}
```

```
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:Sophie ?p ?v
}
```

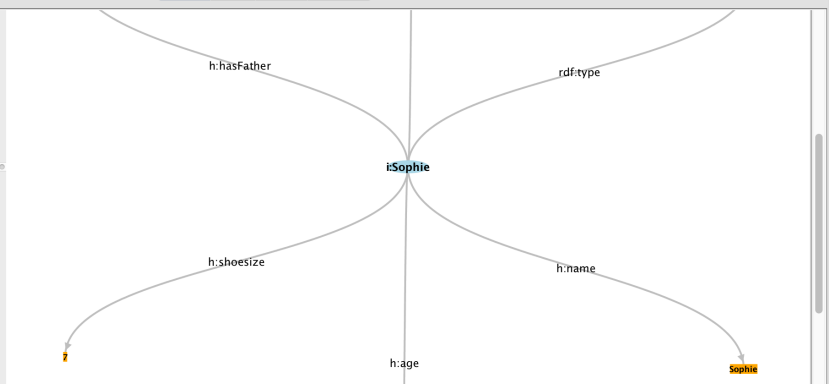
Graph XML Table Validate

?v	?p
29.0	http://www.inria.fr/2007/09/11/humans.rdfs#age
Christophe	http://www.inria.fr/2007/09/11/humans.rdfs#hasFather
Vidhya	http://www.inria.fr/2007/09/11/humans.rdfs#hasFriend
Sophie	http://www.inria.fr/2007/09/11/humans.rdfs#name
7	http://www.inria.fr/2007/09/11/humans.rdfs#shoesize
http://www.inria.fr/2007/09/11/humans.rdfs#Person	http://www.w3.org/1999/02/22-rdf-syntax-ns#type

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX i: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
CONSTRUCT {?x ?p ?v} where {
  ?x ?p ?v; h:name 'Sophie'
}
```

Graph XML Table Validate

```
    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;
```



Question 11

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

Query:

```
I used the filter to avoid duplicates

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

SELECT  ?x ?y ?s WHERE {
      ?x h:shirtsize ?s.
      ?y h:shirtsize ?s.
      FILTER(?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#>

CONSTRUCT {?x h:seeAlso ?y}
WHERE {
      ?x h:shirtsize ?s.
      ?y h:shirtsize ?s.
      FILTER(?x != ?y)
}
```

3. Change the query into an insert.

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

INSERT {?x h:seeAlso ?y}
WHERE {
            ?x h:shirtsize ?s.
            ?y h:shirtsize ?s.
            FILTER(?x != ?y)
      }
```

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

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

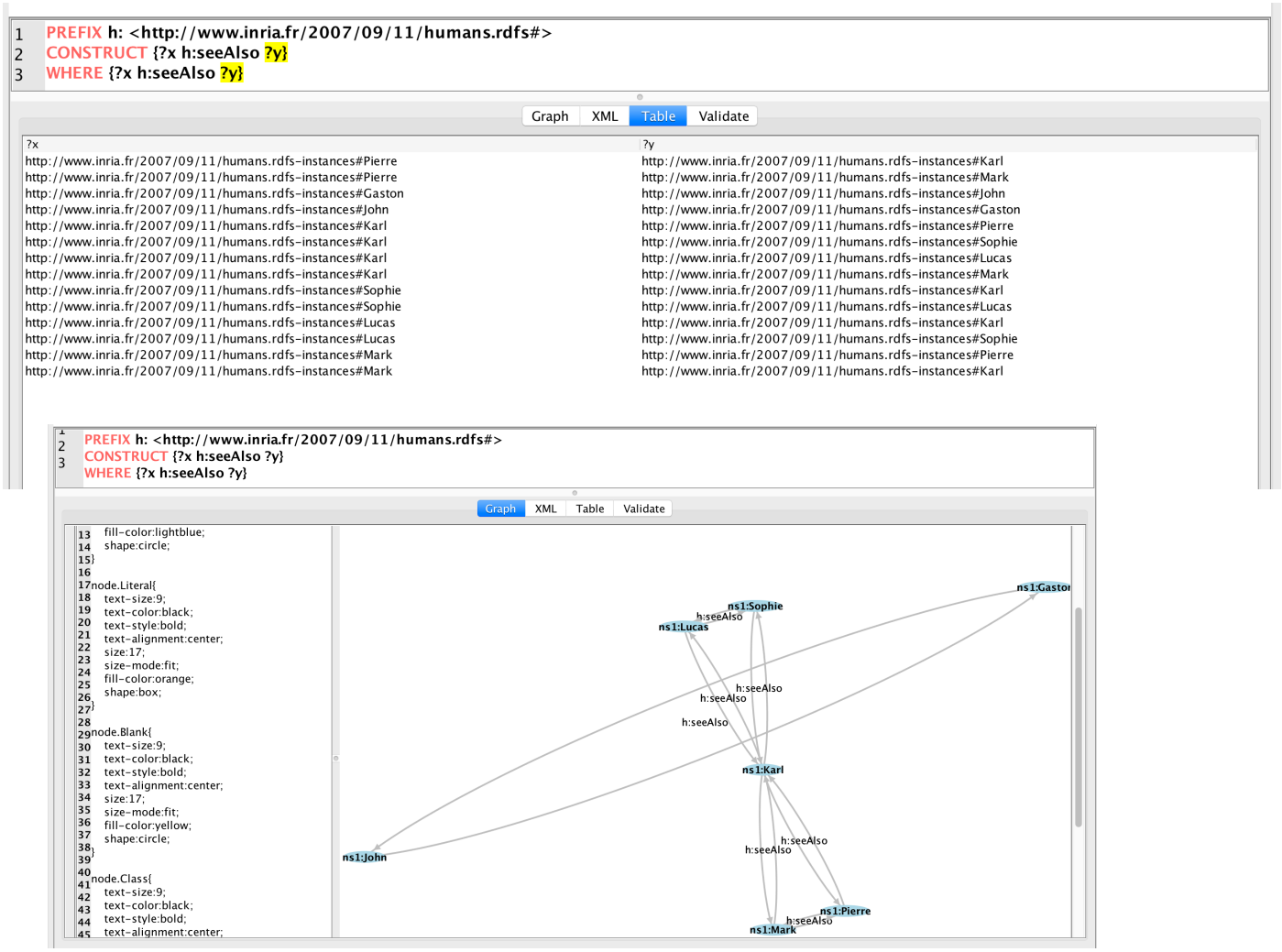
Query:

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

INSERT {?x h:seeAlso ?y}
WHERE {
      ?x h:shoesize ?shoe.
      ?y h:shoesize ?shoe.
      FILTER(?x != ?y)
}
```

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

screenshot:

Note that my newly created person Sophie is in the results.



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

Query:

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

CONSTRUCT {?x h:seeAlso ?y}
WHERE {?x h:seeAlso+ ?y}
```

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

```
No result
```

Question 12

1. Find the largest shoe size

Query:

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

SELECT (MAX(?shoe) as ?max) WHERE {
      ?x h:shoesize ?shoe
}

>> 14
```

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 ?name ?max WHERE {
      {SELECT (MAX(?shoe) as ?max)
      WHERE {?x h:shoesize ?shoe}
      }
      ?bigfoot h:name ?name;
            h:shoesize ?max
}
```

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(?shoe) AS ?average_shoe_size )
WHERE {?x h:shoesize ?shoe}
```

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 ((SUM(?shoe))/(COUNT(?shoe)) AS ?average_shoe_size )
WHERE {?x h:shoesize ?shoe}
```

Question 13

Find couples without children

Query:

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

SELECT * WHERE {
      ?x h:hasSpouse ?y
      MINUS {?x h:hasChild ?v}
      MINUS {?y h:hasChild ?w}
}
```

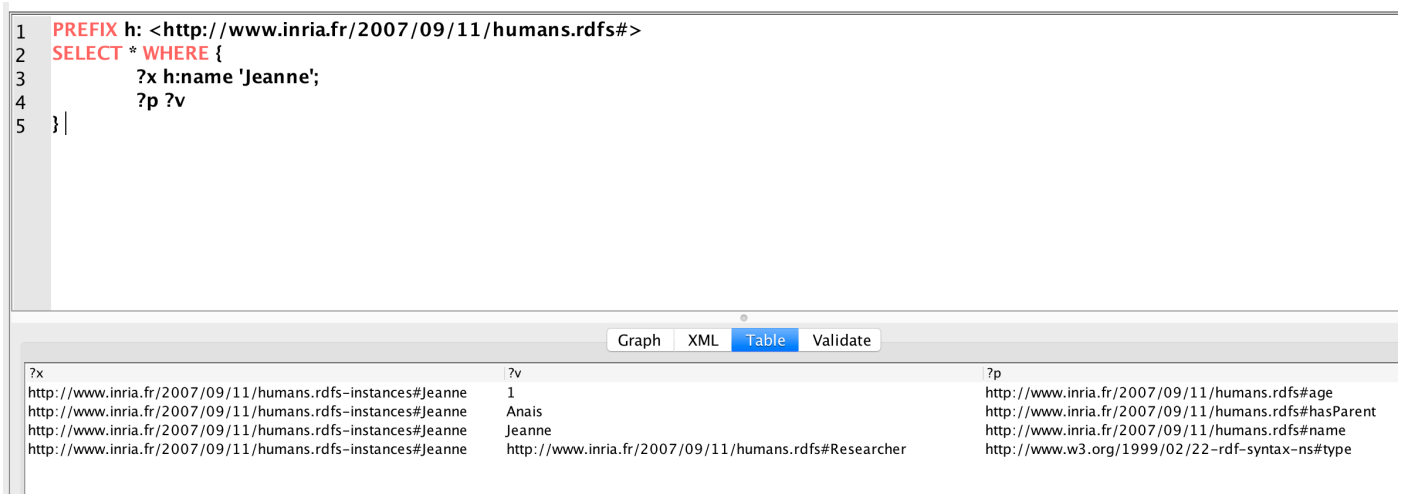
Question 14

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:Jeanne h:hasParent 'Anais';
                 h:age 1;
                 h:name 'Jeanne';
                 a h:Researcher.
}
```

Screenshot result:



```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
SELECT * WHERE {
        ?x h:name 'Jeanne';
        ?p ?v
}
```

Graph | XML | Table | Validate

?x	?v	?p
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jeanne	1	http://www.inria.fr/2007/09/11/humans.rdfs#age
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jeanne	Anais	http://www.inria.fr/2007/09/11/humans.rdfs#hasParent
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jeanne	Jeanne	http://www.inria.fr/2007/09/11/humans.rdfs#name
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jeanne	http://www.inria.fr/2007/09/11/humans.rdfs#Researcher	http://www.w3.org/1999/02/22-rdf-syntax-ns#type

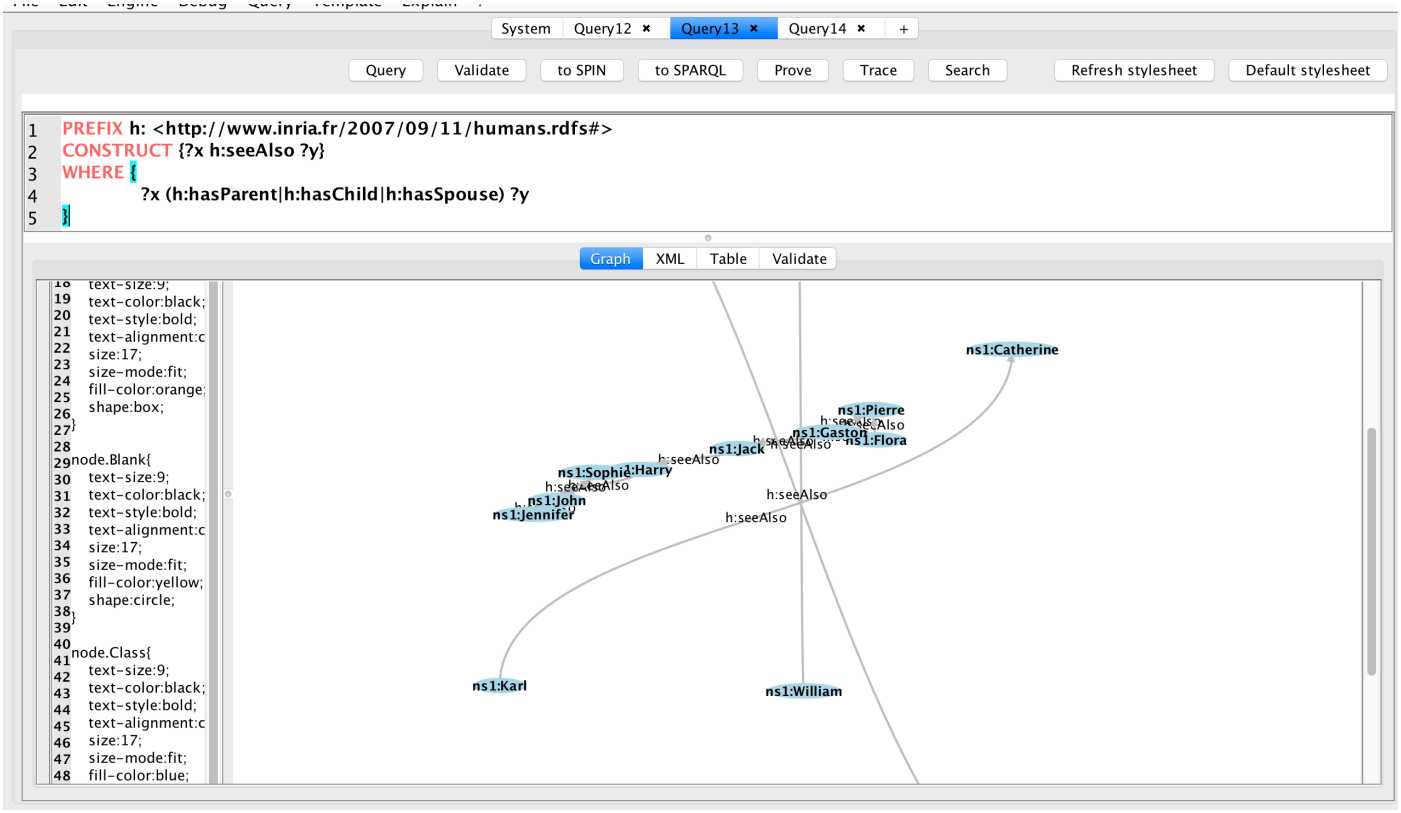
Question 15

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

query:

```
I deleted the person created above (Jeanne).

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

CONSTRUCT {?x h:seeAlso ?y}
WHERE {
      ?x (h:hasParent | h:hasChild | h: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://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

It uses a remote access to a SPARQL endpoint, the French DBpedia, and select 20 married couples (resources that are spouses and both have a name). It then use the construct clause to show a visualization of these resources

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
  }
}
```

screenshot:

```
1
2  prefix h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
3  select * {?x h:hasSpouse ?y}
```

Graph | XML | Table | Validate

?x	?y
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve	http://www.inria.fr/2007/09/11/humans.rdfs-instances#David
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine
http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie
http://fr.dbpedia.org/resource/Guillaume_IV_(roi_du_Royaume-Uni)	http://fr.dbpedia.org/resource/Adélaïde_de_Saxe-Meiningen
http://fr.dbpedia.org/resource/Isabelle_Ire_de_Jérusalem	http://fr.dbpedia.org/resource/Amaury_II_de_Lusignan
http://fr.dbpedia.org/resource/Ariane_de_Rothschild	http://fr.dbpedia.org/resource/Benjamin_de_Rothschild
http://fr.dbpedia.org/resource/Maximilien_Ier_du_Saint-Empire	http://fr.dbpedia.org/resource/Blanche-Marie_Sforza
http://fr.dbpedia.org/resource/Philibert_Ier_de_Savoie	http://fr.dbpedia.org/resource/Blanche-Marie_Sforza
http://fr.dbpedia.org/resource/Bruce_Paltrow	http://fr.dbpedia.org/resource/Blythe_Danner
http://fr.dbpedia.org/resource/Dinah_le_teckel	http://fr.dbpedia.org/resource/Butch_le_bouledogue
http://fr.dbpedia.org/resource/Butch_le_bouledogue	http://fr.dbpedia.org/resource/Dinah_le_teckel
http://fr.dbpedia.org/resource/Jérôme_Bonaparte	http://fr.dbpedia.org/resource/Catherine_de_Wurtemberg
http://fr.dbpedia.org/resource/Gus_Glouton	http://fr.dbpedia.org/resource/Clara_Cluck
http://fr.dbpedia.org/resource/Panchito_Pistoles	http://fr.dbpedia.org/resource/Clara_Cluck
http://fr.dbpedia.org/resource/Wolfgang_Amadeus_Mozart	http://fr.dbpedia.org/resource/Constance_Mozart
http://fr.dbpedia.org/resource/Antoinette_Feuerwerker	http://fr.dbpedia.org/resource/David_Feuerwerker
http://fr.dbpedia.org/resource/Pluto_(Disney)	http://fr.dbpedia.org/resource/Dinah_le_teckel
http://fr.dbpedia.org/resource/Pluto_(Disney)	http://fr.dbpedia.org/resource/Fifi_le_pékinois
http://fr.dbpedia.org/resource/Matilda_Picsou	http://fr.dbpedia.org/resource/Donald_Dingue
http://fr.dbpedia.org/resource/Nicolae_Ceaușescu	http://fr.dbpedia.org/resource/Elena_Ceaușescu
http://fr.dbpedia.org/resource/Niall_Frossach	http://fr.dbpedia.org/resource/Flaithbhertach
http://fr.dbpedia.org/resource/Hannelore_Schmidt	http://fr.dbpedia.org/resource/Helmut_Schmidt
http://fr.dbpedia.org/resource/Irène_Ovtchinnikova	http://fr.dbpedia.org/resource/Pierre_de_Grèce

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: http://wimmics.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 may have to update the files Jen.rdf and Jen.ttl to use your ontology.

Explanation (for myself):
Data and actual persons in rdf data.
Schema with list of properties and classes in rdfs.
prefix in rdf should match the rdfs base URI.

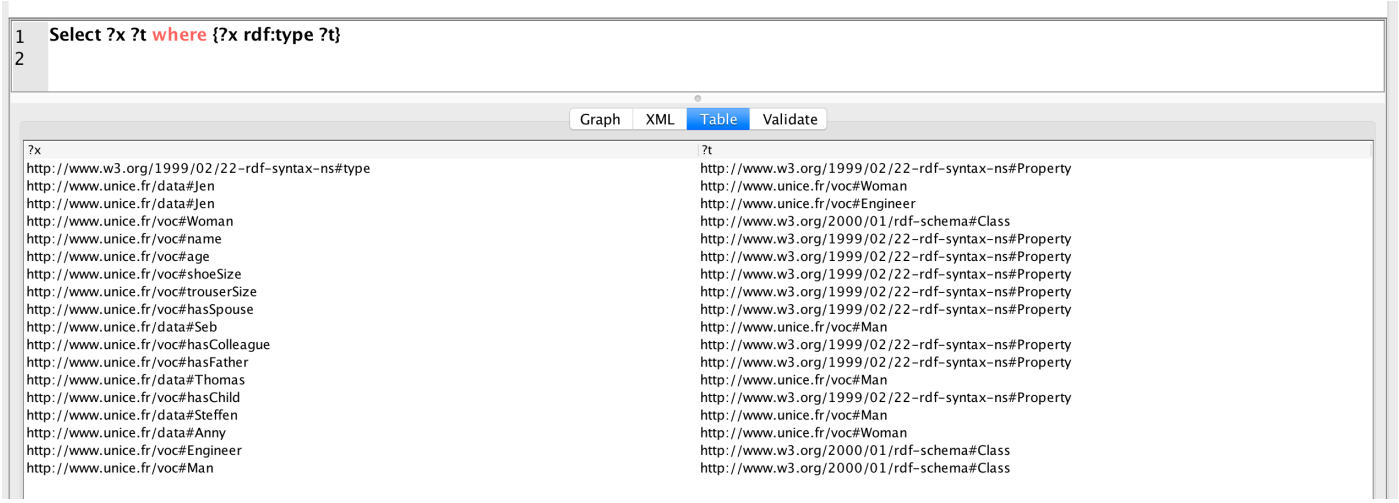
Your schema:

```
<rdf:RDF
  xml:base="http://www.unice.fr/voc"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">


  <rdf:Property rdf:ID="name">
  </rdf:Property>

  <rdf:Property rdf:ID="shoeSize">
  </rdf:Property>

  <rdf:Property rdf:ID="trouserSize">
  </rdf:Property>

  <rdf:Property rdf:ID="age">
  </rdf:Property>

  <rdf:Property rdf:ID="hasChild">
  </rdf:Property>

  <rdf:Property rdf:ID="hasFather">
  </rdf:Property>

   <rdf:Property rdf:ID="hasSpouse">
  </rdf:Property>

 <rdf:Property rdf:ID="hasColleague">
  </rdf:Property>


  <rdfs:Class rdf:ID="Man">
  </rdfs:Class>

  <rdfs:Class rdf:ID="Woman">
  </rdfs:Class>

  <rdfs:Class rdf:ID="Engineer">
  </rdfs:Class>
```

```
</rdf:RDF>
```

- 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:



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

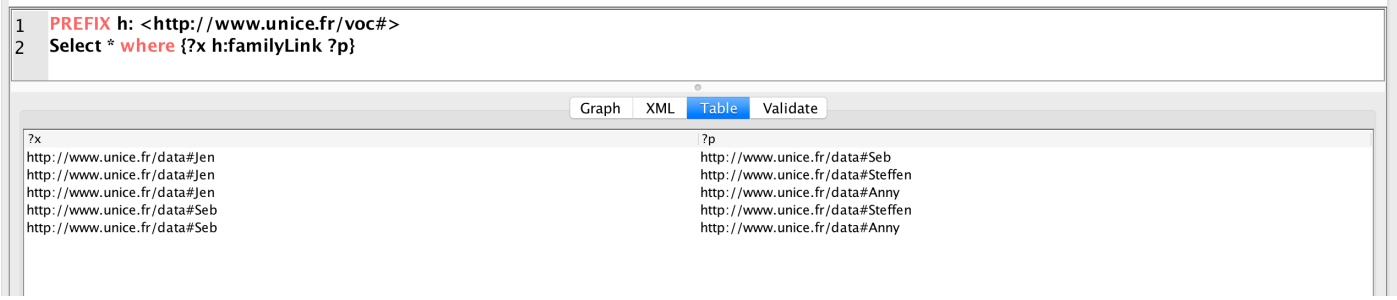
Screenshot:

Explanation:

All the persons above belong to the class Man or Woman, which are both subclasses of Human. Thus the system automatically categorizeS them as human.

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

Screenshot:

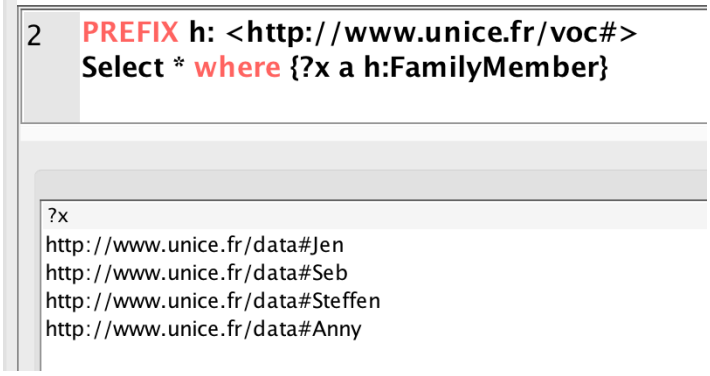


Explanation:

All the people above have a hasSpouse or hasChild property. As a result, the system applied the property of the parent class, 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:



Explanation:

As I specified that the property familyLink was linking FamilyMembers (both domain and range), all resources with the property familyLink were added to the class FamilyMembers.

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?

"http://www.inria.fr/2007/09/11/humans.rdfs" with xml:base="URI"

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?

Classes and most of the other terms are associated with the default namespace: "http://www.w3.org/2000/01/rdf-schema#"

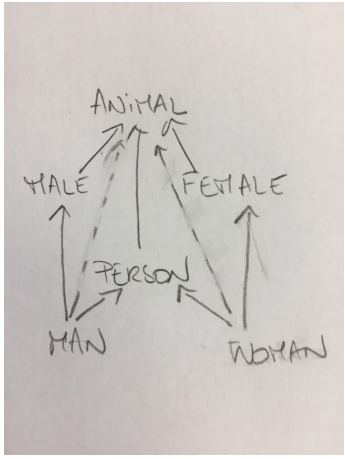
The Property term is associated with the rdf namespace: "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

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

All as they are no restrictions or other types of constraints.

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

Drawing of hierarchy:



I drew the transitive parent link between Man, Woman and Animal with - - - >

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 * 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 {
      ?childclass rdfs:subClassOf ?parentclass
}
```

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:comment ?comment
      FILTER(lang(?comment) != 'en')}
      UNION
      {h:shoesize rdfs:label ?label
      FILTER(lang(?label) != 'en')}
}
```

answers:

?comment	?label
taille, exprimée en points, des chaussures d'une personne.@fr	
	pointure@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 * WHERE {
      {SELECT (?x as ?p) WHERE {?x rdfs:label "personne"@fr}}
      ?p rdfs:label ?label.
      FILTER(lang(?label) = 'fr')
}
```

answers:

?p	?label
http://www.inria.fr/2007/09/11/humans.rdfs#Person	"homme"@fr
http://www.inria.fr/2007/09/11/humans.rdfs#Person	"personne"@fr
http://www.inria.fr/2007/09/11/humans.rdfs#Person	"être humain"@fr
http://www.inria.fr/2007/09/11/humans.rdfs#Person	"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:

```
I kept the reference to the resources/properties and to the label, in addition
to the comments in the first query.

In the second I just kept the reference to the properties and the comments.

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT * WHERE {
      {SELECT DISTINCT (?p as ?r) ?label WHERE {
            ?p rdfs:label ?label.
            FILTER(CONTAINS(?label, 'size'))}
      }
      ?r rdfs:comment ?comment
}

PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT * WHERE {
      {SELECT DISTINCT (?p as ?r) WHERE {
            ?p rdfs:label ?label.
            FILTER(CONTAINS(?label, 'size'))}
      }
      ?r rdfs:comment ?comment
}
```

answers:

```
From the second query:
```

?r	?comment
http://www.inria.fr/2007/09/11/humans.rdfs#shoesize	"express in some way the approximate length of the shoes for a person."@en
http://www.inria.fr/2007/09/11/humans.rdfs#shoesize	"taille, exprimée en points, des chaussures d'une personne."@fr
http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize	"express in some way the approximate dimensions of the shirts of a person."@en
http://www.inria.fr/2007/09/11/humans.rdfs#shirtsize	"dimensions approximatives des chemises portées par une personne."@fr
http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize	"express in some way the approximate dimensions of the trousers of a person."@en
http://www.inria.fr/2007/09/11/humans.rdfs#trouserssize	"dimensions approximatives des pantalons portés par une personne."@fr

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

query:

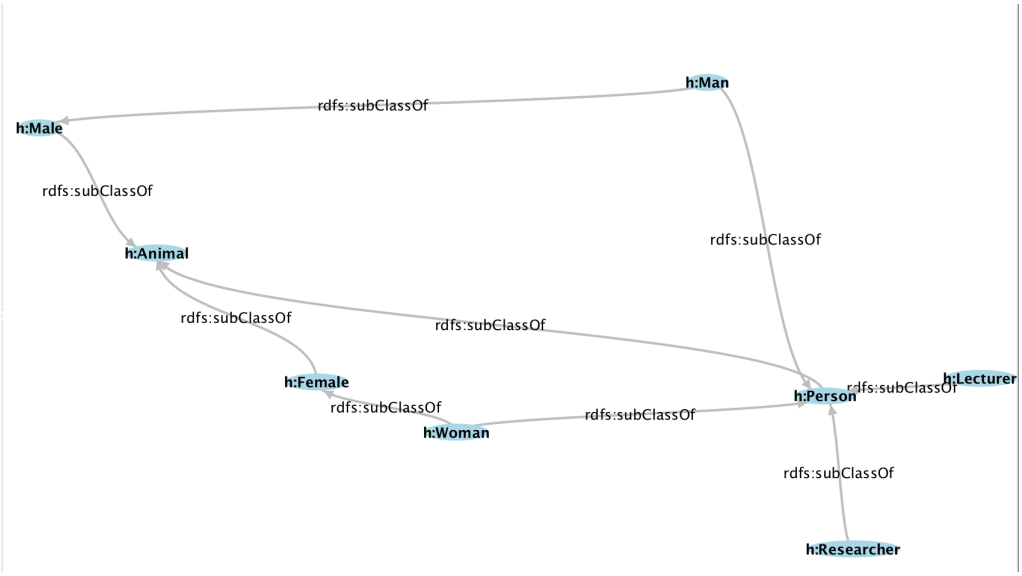
```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT * WHERE {?x ?p h:Person.
          FILTER(?p != rdfs:subClassOf)
          }
```

7. Rebuild the hierarchy of Classes (CONSTRUCT) considering only the classes in the humans.rdfs schema

query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT {?c2 ?p ?c1}
WHERE {?c1 a rdfs:Class.
     ?c2 a rdfs:Class;
      ?p ?c1
     }
```

screenshot:



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

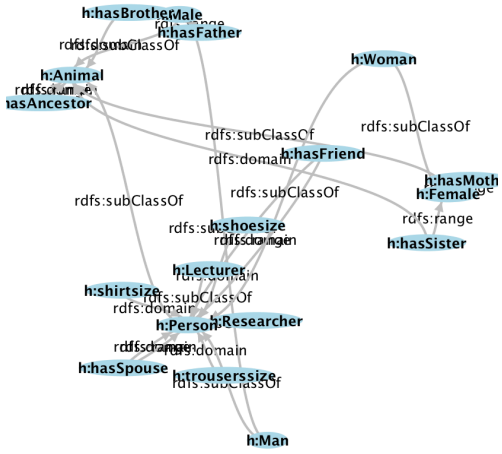
query:

```
PREFIX h: <http://www.inria.fr/2007/09/11/humans.rdfs#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

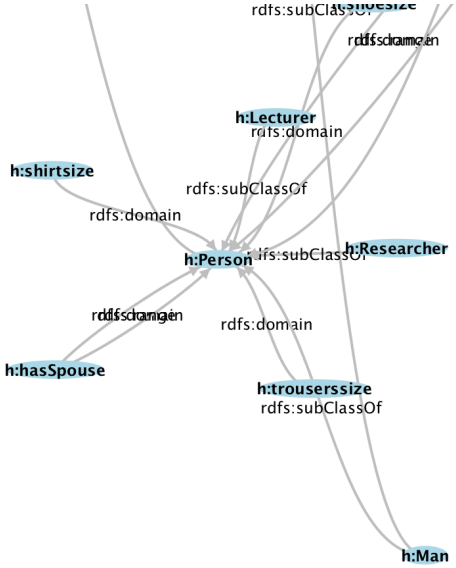
```
CONSTRUCT {?parent ?q ?child.
           ?p rdfs:domain ?cdomain.
           ?p rdfs:range ?crange. }
WHERE {
      {?child a rdfs:Class.
      ?parent a rdfs:Class;
       ?q ?child
      }
      UNION
      {?p rdfs:domain ?cdomain}
      UNION
      {?p rdfs:range ?crange}
      }
```

screenshot:

Overview



zoom



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

SELECT * WHERE {
  ?x  a rdfs: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
```

Some people were in subclasses of Person, or had properties linked to being in the class Person. When we loaded the schema, they were added to the class Person.

Question 2

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

Query:

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

SELECT * WHERE {
  ?m a rdfs:Male;
     rdfs:hasSpouse ?w
}
```

Number of results and explanation:

I only get one answer, Harry and Sophie. This is because we haven't defined that hasSpouse was symmetric, and that for the other couples, it must have been declared as <womanName> hasSpouse <manName>, and/or for some resources we don't know whether they are Man or not.

2. In the data declare that Lucas has for 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:

```
     <hasFather rdf:resource="#Karl"/>
```

Number of results before and after and explanation:

2 results (1 more than before). The new line is Karl and Catherine. We knew already that Karl hasSpouse Catherine. By adding the new line in rdf, now it has been inferred that Karl is a male, which explains why we get the new result.

Question 3

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

Query:

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

SELECT * WHERE {
     ?x a rdfs:Lecturer, ?type
}
```

Number of results and your explanation:

Including the type Lecturer, I get 8 results (and 6 otherwise).

Lecturer and Researcher (Laura only) were declared in the data.

While we didn't directly define in the rdf that Eve, Laura were Persons, a Lecturer is a subclass of Person and so it was inferred they are both Persons and thus both Animals.
We have Catherine hasMother Laura from which is deduced Laura is a Female.

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

```
SELECT * WHERE {
      ?x a rdfs:Male, rdfs:Person
}
```

Your explanation of the result:

```
We get 7 results, including Jack. Jack was defined as a Man, which is a
subclass of Male and Person, and that's why we get him in the results.
```

Question 4

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

Query:

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

SELECT * WHERE {

      ?x rdfs:hasAncestor ?ancestor

}
```

Your explanation of the result:

```
I get 5 results.

While hasAncestor was never used in the rdf data, it is a superclass of
hasParent, which is a superclass of hasMother, hasFather and as a result, every
resource that has a hasMother, hasFather, hasParent relation has a hasAncestor
relation.
```

Question 5

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

Query:

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

SELECT  * WHERE {
      ?x rdfs:hasSpouse ?y
      optional {?y rdfs:hasSpouse ?x}
      optional {?x rdfs:hasChild ?child1}
      optional {?y rdfs:hasChild ?child2}
}
```

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

Query:

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

CONSTRUCT {?x rdfs:hasSpouse ?y; rdfs:hasChild ?child1.
           ?y rdfs:hasSpouse ?x; rdfs:hasChild ?child2}

WHERE {
      ?x rdfs:hasSpouse ?y
      optional {?y rdfs:hasSpouse ?x}
      optional {?x rdfs:hasChild ?child1}
      optional {?y rdfs:hasChild ?child2}
}
```

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">
  <range rdf:resource="#Person"/>
  <domain rdf:resource="#Person"/>
  <label xml:lang="en">is older than</label>
  <label xml:lang="fr">est plus age</label>
</rdf:Property>
```

Query:

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

INSERT {?x rdfs:olderThan ?y}
WHERE {
      ?x rdfs:age ?xage.
      ?y rdfs:age ?yage.
      FILTER(?xage > ?yage)
}
```

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

Query:

```
I included both ages in the query for verification purposes.

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

SELECT *
WHERE {?x rdfs:olderThan ?y;
           rdfs:age ?xage.
      ?y rdfs:age ?yage.
      MINUS {?x rdfs:olderThan ?z.
            ?z rdfs:olderThan ?y}
}
```

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 inst: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
PREFIX h: <http://www.w3.org/2000/01/rdf-schema#>

SELECT  * WHERE {
      {SELECT  DISTINCT (?p as ?prop) WHERE {
      ?p h:label ?label;
      FILTER(CONTAINS(?label,'size'))
}}
      inst:John ?prop ?value;
}
```

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:

```
If I understood the question the right way:

PREFIX h: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX inst: <http://www.inria.fr/2007/09/11/humans.rdfs-instances#>
```

```
SELECT *
WHERE {
      {{SELECT *
      WHERE {?y h:label ?label1
      FILTER(lang(?label1)='fr')}}.
      inst:Laura a ?y.}
      UNION
      {
      {SELECT *
      WHERE {?x h:label ?label2
      FILTER(lang(?label2)='fr')}}.
      inst:Laura ?x ?y.
      }
}
```

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: http://wimmics.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` .

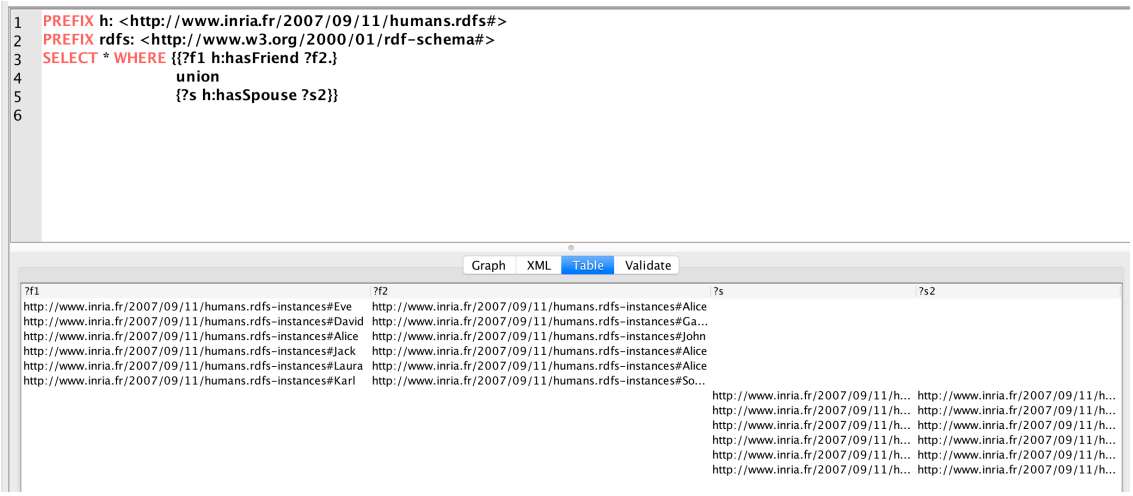
Code added to the schema:

```
In the RDF <>:
xmlns:owl = "http://www.w3.org/2002/07/owl#"
```

```
Replaced rdf:Property by owl:SymmetricProperty:
<owl:SymmetricProperty rdf:ID="hasFriend">
      (…)
</owl:SymmetricProperty>

<owl:SymmetricProperty rdf:ID="hasSpouse">
      (…)
</owl:SymmetricProperty>
```

Query:

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

SELECT * WHERE {
           {?f1 h:hasFriend ?f2.}
           UNION
           {?s1 h:hasSpouse ?s2}}

Result before addition to the schema: 12 results
```



Result after addition to the schema:

24 = 12*2, thanks to the addition of the symmetric properties.

?f1	?f2	?s1	?s2
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#David	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#David		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Alice		
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie		
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie

		http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve	http://www.inria.fr/2007/09/11/humans.rdfs-instances#David
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#David	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura	http://www.inria.fr/2007/09/11/humans.rdfs-instances#William
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jennifer	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine
		http://www.inria.fr/2007/09/11/humans.rdfs-instances#William	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura

Explanation:

```
Each time we had a one way relationship of type hasFriend, hasSpouse, we now have the same relationship the other way around.
```

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

Code added to the schema:

```
  <owl:inverseOf rdf:resource="#hasParent" />
```

Query:

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

```
SELECT * WHERE {
  ?x h:hasChild ?y
}
```

Result before addition to the schema:

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry
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:

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas

Explanation:

```
For all the resources that had a hasParent relation with other resources, the
latter now have a hasChild relation with the former.
```

3. Declare `hasAncestor` as transitive property.

Code added to the schema:

```
<owl:TransitiveProperty rdf:ID="hasAncestor">
      (…)
</owl:TransitiveProperty>
```

Query:

```
Note to self: verify that Engine> all rules are checked.

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

```
SELECT * WHERE {
  ?x h:hasAncestor ?y
}
```

Result before addition to the schema:

```
5:
```

http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl
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:

```
18 results
```

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#John	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Harry
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs-instances#John
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Sophie
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Mark	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Jack	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Flora
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine

http://www.inria.fr/2007/09/11/humans.rdfs-instances#Lucas	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Karl
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Catherine	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura

Explanation:

```
hasParent is a subclass of hasAncestor, and hasParent is now an inverse
property of hasChild. Also, hasParent is the superclass of hasMother,
hasFather. Hence a lot of relationships are added once we load the owl.
```

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:

```
In the Male class in the schema:
<owl:disjointWith rdf:resource="#Female"/>

In the rdf data, in Pierre <>
<rdf:type rdf:resource="&humans;#Female"/>
```

Query:

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

SELECT ?p ?y WHERE {
      ?x ?p ?y.
      ?y h:name "Pierre"}
```

Result before addition to the schema:

?p	?y
http://www.inria.fr/2007/09/11/humans.rdfs#hasChild	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.inria.fr/2007/09/11/humans.rdfs#hasChild	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre

Result after addition to the schema:

?p	?y
http://spinrdf.org/sp#violationRoot	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.inria.fr/2007/09/11/humans.rdfs#hasChild	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.inria.fr/2007/09/11/humans.rdfs#hasChild	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre
http://www.w3.org/2002/07/owl#sameAs	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Pierre

Explanation:

```
We get a notice that there is a violation root on Pierre, coming from him being
defined as both Male and Female while the two classes are disjoints.
```

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="#Lecturer"/>
    <owl:Class rdf:about="#Researcher"/>
  </owl:intersectionOf>
</owl:Class>
```

Query:

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

SELECT * WHERE {
     ?x a h:Professor}
```

Result before addition to the schema:

```
None
```

Result after addition to the schema:

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

Explanation:

```
Once we load the OWL, the Professor class, which is comprised of every resource
that is both a Lecturer and a Researcher is created. Laura is both a Lecturer
and a Researcher, and thus a Professor according to the new OWL rule.
```

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

Query:

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

SELECT * WHERE {
     ?x a h:Academic}
```

Result before addition to the schema:

```
None
```

Result after addition to the schema:

?x
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Eve
http://www.inria.fr/2007/09/11/humans.rdfs-instances#David
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Gaston
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Laura

Explanation:

```
Once we load the OWL, the Academic class, which is comprised of every resource
that is either a Lecturer and a Researcher is created. The 4 resources above
match one of the criteria, and are thus Academic according to the new OWL rule.
```

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):

New restriction within Professor class

```
<owl:Class rdf:ID="Professor">
  (…previous_code…)
  <subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#mainEmployer" />
      <owl:allValuesFrom rdf:resource="#University" />
    </owl:Restriction>
  </subClassOf>
</owl:Class>
```

New classes and property

```
<owl:Class rdf:ID="Organization">
</owl:Class>

<owl:Class rdf:ID="University">
  <subClassOf rdf:resource="#Organization"/>
</owl:Class>

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

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

```
<Researcher rdf:about="#Laura">
  (…previous_code…)
  <mainEmployer>Inria</mainEmployer>
</Researcher>
```

Query:

```
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:Laura h:mainEmployer ?employer
     OPTIONAL{?employer a ?emp_type}}
```

Result before addition to the schema:

?employer	?emp_type
Unice	

Result after addition to the schema:

?employer	?emp_type
Unice	http://www.inria.fr/2007/09/11/humans.rdfs#University
Unice	http://www.inria.fr/2007/09/11/humans.rdfs#Organization
Unice	http://www.w3.org/2002/07/owl#Thing

Explanation:

With a restriction, we've defined that all professors have for mainEmployer a university. Thus, as Laura as for mainEmployer Unice and Laura is a professor, Unice is defined as a university. As University is a subclass of Organization (and because the range of mainEmployer is Organization), Unice is defined as an Organization.

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:

```
Added to owl:

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

```
Added to rdf:

<Animal rdf:about="#Brendan">
     <hasParent rdf:resource="#Terri"/>
</Animal>

<Woman rdf:about="#Terri">
</Woman>
```

Query:

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


SELECT * WHERE {
     ?x h:hasParent ?y
     MINUS{?x a h:Person}

}
```

Result before addition to the schema:

?x	?y
http://www.inria.fr/2007/09/11/humans.rdfs-instances#Brendan	http://www.inria.fr/2007/09/11/humans.rdfs-instances#Terri

Result after addition to the schema:

?x	?y

Explanation:

Before adding the schema, we get one result, Brendan (created for testing this) who has a parent who is a woman, but is not a person. After, and due to the

restriction, anyone who has a woman parent belongs to an anonymous class equivalent to the Person class, and thus we don't get any results anymore. Brendan is now a person.

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* <SEE TABLE BELOW/>
2. *owl:unionOf* <SEE TABLE BELOW/>
3. *owl:intersectionOf* <YOUR EXAMPLE HERE/>
4. *owl:complementOf* <YOUR EXAMPLE HERE/>
5. *owl:disjointWith or owl:AllDisjointClasses or owl:disjointUnionOf* <YOUR EXAMPLE HERE/>
6. *owl:ObjectProperty* <YOUR EXAMPLE HERE/>
7. *owl:DatatypeProperty* <YOUR EXAMPLE HERE/>
8. *owl:SymmetricProperty or owl:AsymmetricProperty* <YOUR EXAMPLE HERE/>
9. *owl:inverseOf* <YOUR EXAMPLE HERE/>
10. *owl:TransitiveProperty* <YOUR EXAMPLE HERE/>
11. *owl:propertyDisjointWith* <YOUR EXAMPLE HERE/>
12. *owl:ReflexiveProperty or owl:IrreflexiveProperty* <YOUR EXAMPLE HERE/>
13. *owl:propertyChainAxiom* <YOUR EXAMPLE HERE/>
14. *owl:FunctionalProperty* <YOUR EXAMPLE HERE/>
15. *owl:InverseFunctionalProperty* <YOUR EXAMPLE HERE/>
16. *owl:hasKey* <YOUR EXAMPLE HERE/>
17. *owl:allValuesFrom* <YOUR EXAMPLE HERE/>
18. *owl:someValuesFrom* <YOUR EXAMPLE HERE/>
19. *owl:hasValue* <YOUR EXAMPLE HERE/>
20. *owl:maxCardinality or owl:minCardinality* <YOUR EXAMPLE HERE/>
21. *owl:qualifiedCardinality* <YOUR EXAMPLE HERE/>

OWL Primitive	Def	OWL code
owl:oneOf	The time can either be past, present or future.	<owl:Class rdf:id="Time"> <owl:oneOf rdf:parseType="Collection"> <owl:Thing rdf:ID="Past"/> <owl:Thing rdf:ID="Present"/> <owl:Thing rdf:ID="Future"/> </owl:oneOf> </owl:Class>

OWL Primitive	Def	OWL code
owl:unionOf	A beverage is either an alcoholic drink or a non alcoholic drink.	`<owl:Class rdf:id="Beverage"> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#NonAlcoholicDrink"/> <owl:Class rdf:about="#AlcoholicDrink"/> </owl:unionOf> </owl:Class>`
owl:intersectionOf	A Ballerina is a woman who dances and who is part of a Ballet crew	`<owl:Class rdf:id="Ballerina"> <owl:intersectionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Woman"/> <owl:Class rdf:about="#Dancer"/> <owl:Class rdf:about="#BalletCrew"/> </owl:intersectionOf> </owl:Class>`
owl:disjointWith	An adult cannot be a children and vice versa	`<owl:Class rdf:ID="Adult"> <owl:disjointWith rdf:resource="#Children"/> </owl:Class>`
owl:ObjectProperty	The property hasEnemy is a relation between two resources (such as Eva and Steve)	`<owl:ObjectProperty rdf:ID="hasFriend"> <owl:propertyChainAxiom rdf:parseType="Collection"> <owl:ObjectProperty rdf:about="#hasEnemy"/> <owl:ObjectProperty rdf:about="#hasEnemy"/> </owl:propertyChainAxiom> </owl:ObjectProperty>`
owl:DatatypeProperty	The property hasWeight can be typed as float	`<owl:DatatypeProperty rdf:ID="hasWeight"> <rdfs:domain rdf:resource="#Person" /> <rdfs:range rdf:resource="&xsd;float"/> </owl:DatatypeProperty>`
owl:SymmetricProperty	If Eva is the cousin of Steve then Steve is the cousin of Eva	`<owl:SymmetricProperty rdf:ID="hasCousin"/>`
owl:inverseOf	If Eva is the teacher of Steve then Steve is the student of Eva.	`<rdf:Property rdf:ID="hasTeacher"> <owl:inverseOf rdf:resource="#hasStudent" /> </rdf:Property>`
owl:TransitiveProperty	If Eva is younger than Steve and Steve is younger than Lou then Eva is younger than Lou.	`<owl:TransitiveProperty rdf:ID="isYoungerThan" />`
owl:propertyDisjointWith	If Eva is tall then Eva isn't small	`<isTall> owl:propertyDisjointWith <isSmall>`
owl:IrreflexiveProperty	You cannot be your own colleague	`<owl:IrreflexiveProperty rdf:about="hasColleague"/>`

OWL Primitive	Def	OWL code
owl:propertyChainAxiom	The enemy of my enemy is my friend	<owl:ObjectProperty rdf:ID="hasFriend"> <owl:propertyChainAxiom rdf:parseType="Collection"> <owl:ObjectProperty rdf:about="#hasEnemy"/> <owl:ObjectProperty rdf:about="#hasEnemy"/> </owl:propertyChainAxiom> </owl:ObjectProperty>
owl:FunctionalProperty	A person can only have 1 birthplace	<owl:FunctionalProperty rdf:ID="birthPlace" />
owl:InverseFunctionalProperty	Two invoices with the same invoice_number are the same	<owl:InverseFunctionalProperty rdf:ID="invoice_number" />
owl:hasKey	Two invoices for the same client on the same date and with the same total are the same.	<owl:Class rdf:ID="Invoice"> <owl:hasKey rdf:parseType="Collection"> <owl:ObjectProperty rdf:about="#client"/> <owl:ObjectProperty rdf:about="#date"/> <owl:ObjectProperty rdf:about="#total"/> </owl:hasKey> </owl:Class>
owl:allValuesFrom	A dolphin only sees in Black and White	<owl:Class rdf:ID="Dolphin"> <subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#sees" /> <owl:allValuesFrom rdf:resource="#GrayScale" /> </owl:Restriction> </subClassOf> </owl:Class>
owl:someValuesFrom	Among the things a healthy person has eaten there must be some vegetables	<owl:Class rdf:ID="Healthy_Person"> <subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#hasEaten" /> <owl:someValuesFrom rdf:resource="#Vegetable" /> </owl:Restriction> </subClassOf> </owl:Class>
owl:hasValue	All dolphins have 1 tail	<owl:Class rdf:ID="Dolphin"> <owl:Restriction> <owl:onProperty rdf:resource="#nbTails" /> <owl:hasValue>1</owl:hasValue> </owl:Restriction> </subClassOf> </owl:Class>

OWL Primitive	Def	OWL code

owl:maxCardinality	A (living) person must have at least 1 kidney	`<owl:Class rdf:ID="Person">`
		`<subClassOf><owl:Restriction>`
		`<owl:onProperty`
		`rdf:resource="#hasKidney" />`
		`<owl:minCardinality>1</owl:minCardinality>`
		`</owl:Restriction></subClassOf>`
		`</owl:Class>`

owl:qualifiedCardinality	A house can only be legally owned by 1 legal agent	`<owl:Class rdf:ID="House">`
		`<rdfs:subClassOf><owl:Restriction>`
		`<owl:onProperty`
		`rdf:resource="#hasOwner" />`
		`<owl:onClass rdf:resource="#LegalAgent"/>`
		`<owl:qualifiedCardinality>1</`
		`owl:qualifiedCardinality>`
		`</owl:Restriction></rdfs:subClassOf>`
		`</owl:Class>`