Linked Data Tutorial

By: Noureddin Sadawi
`http://people.brunel.ac.uk/~csstnns`

05 Feb 2014

1 Overview

In this short tutorial we are going to see how we can create and manipulate semantic data using ontologies and datastores. We are going to use Protege to design a simple ontology, insert some initial instances in this ontology and save the output as RDF/XML format. Then we are going to make this file available online. You can place it in your university web-space. I am going to place it on `http://people.brunel.ac.uk/~csstnns/university.owl`. After that, we are going to import this ontology to Apache's Jena Fuseki project and run some simple SPARQL queries (I have recorded this tutorial and a link to the videos can be found here: `http://people.brunel.ac.uk/~csstnns/tutorials.html`).

Note: I am using Protege 4.3.0 (build 304) in Ubuntu Linux so the version you are using might not be the same. However, but the steps we are going to follow should not be very different.

2 Building the Ontology

As Figure 1 shows, our simple ontology will model a university (or a department in a university). We will represent people at this university by two classes, Lecturer and Student.

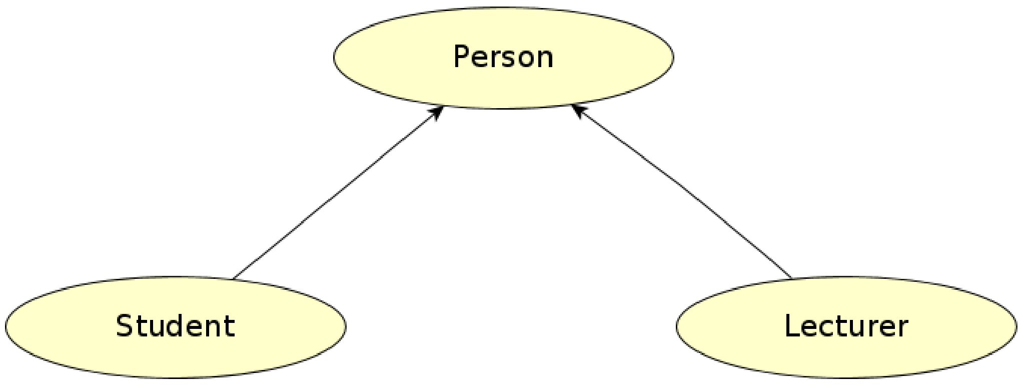


Figure 1: Classes Lecturer and Student

The university offers a number of Modules in Computer Science and Mathematics as summarised in Figure 2. Each Lecturer teaches one or more modules and each Student can study one or more modules.

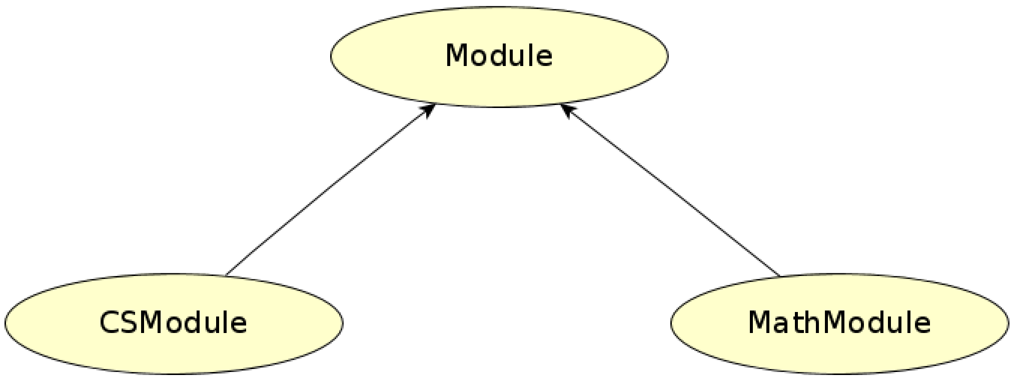


Figure 2: Classes CSModule and MathModule

2.1 Creating the Classes

- Start protege and start a new Ontology
- Enter an Ontology IRI. I am using the following IRI: `http://people.brunel.ac.uk/~csstnns/university.owl`
- Save it in your local file as university.owl
- Go to the *Classes* tab
- The empty class tree contains one class called *Thing*, which is superclass of everything
- Create class *Person* as subclass of *Thing*
- Create *Lecturer* and *Student* as the subclasses of *Person*
- Create class *Module* as subclass of *Thing*

- Create *CSModule* and *MathModule* as the subclasses of *Module*
- Your classes should look like Figure 3
- Make all classes *Disjoint with* each other. Do you know why?

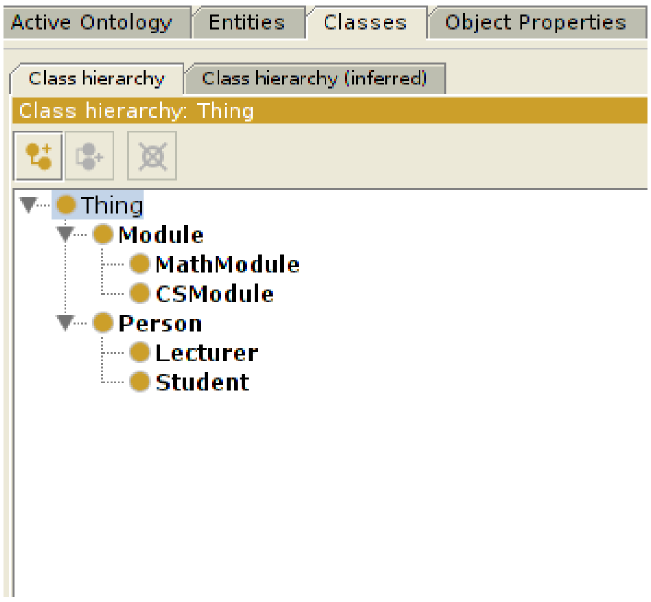


Figure 3: Protege - Basic Classes in our Ontology

2.2 Creating the Object Properties

As you know, object properties describe relationships between two instances (individuals). They link individuals from a *domain* to individuals a *range*. OWL uses domain and range as axioms in reasoning. We will create object properties for our ontology as follow:

- Switch to the *Object Properties* tab
- Use *Add subProperty* button to create a new object property (notice that all object peroperties are subproperties of topObjectProperty)
- Create a new object property *studies* and select Student and Module as its domain and range respectively (Figure 4)

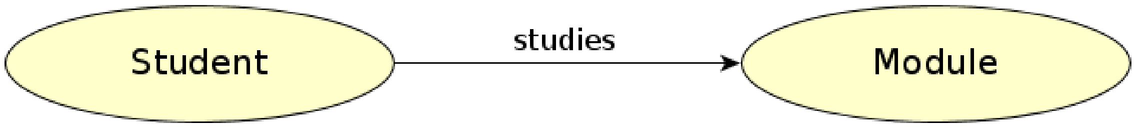


Figure 4: The *studies* Property/Relationship

- Create another new object property *teaches* and select Lecturer and Module as its domain and range respectively (Figure 5)

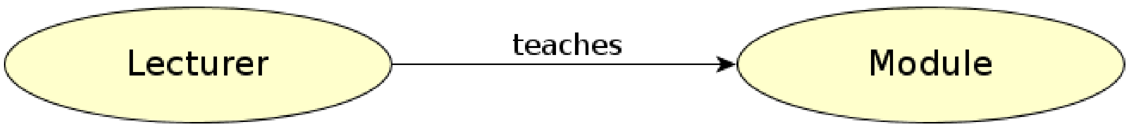


Figure 5: The *teaches* Property/Relationship

- Your object properties should look like Figure 6

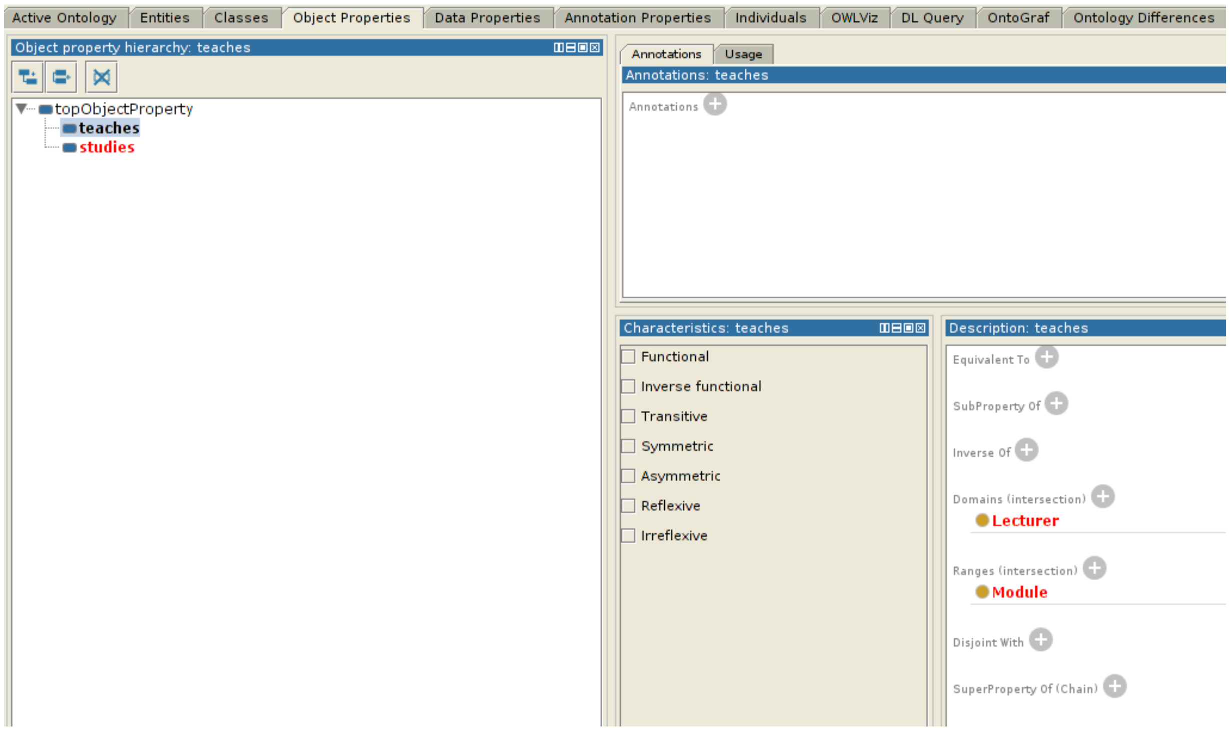


Figure 6: Protege - Object Properties in our Ontology

2.3 Adding Data Properties

In section 2.2 we mentioned that object properties describe relationships between two individuals. Data properties describe relationships between instances (individuals) and data values. We will add some data properties to our ontology as follow:

- Switch to the *Data Properties* tab
- Use *add sub property* button to create a new data property (notice that all data peroperties are subproperties of topDataProperty)
- Create a new data property *first_name* and select *Person* and *string* as its domain and range respectively
- Create a new data property *last_name* and select *Person* and *string* as its domain and range respectively

- Create another new data property *staffID* and select *Lecturer* and *integer* as its domain and range respectively
- Create another new data property *studentID* and select *Student* and *integer* as its domain and range respectively
- Your data properties should look like Figure 7

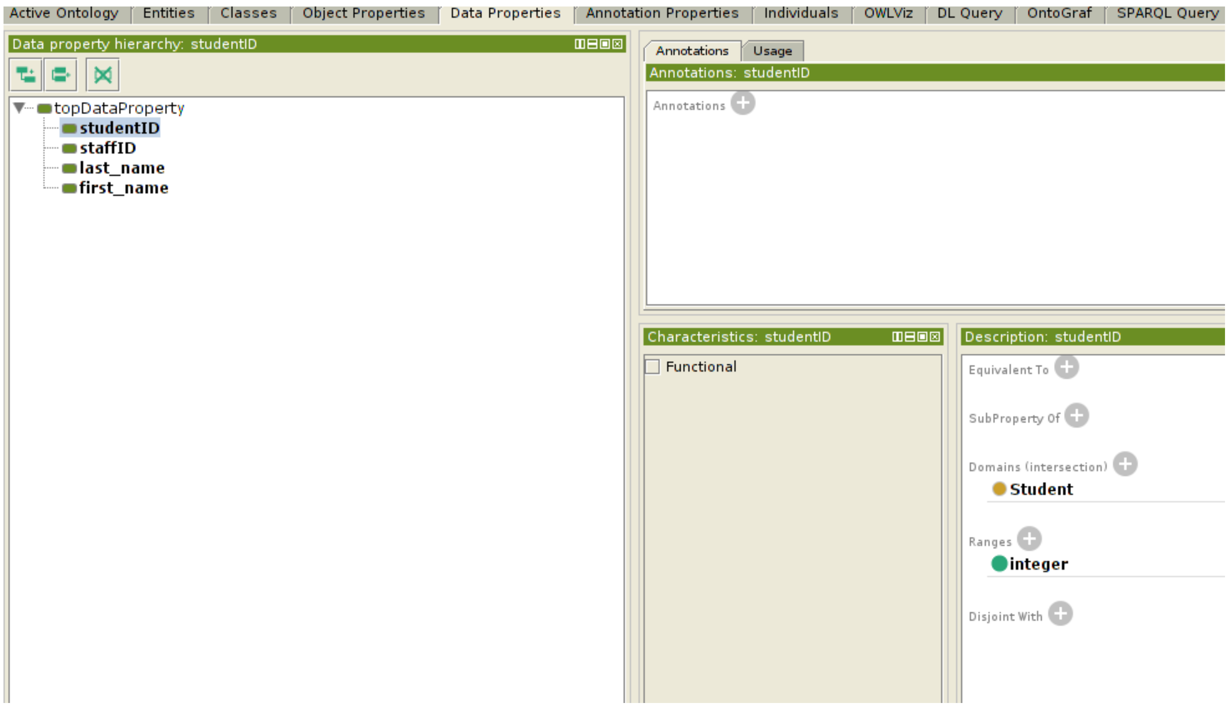


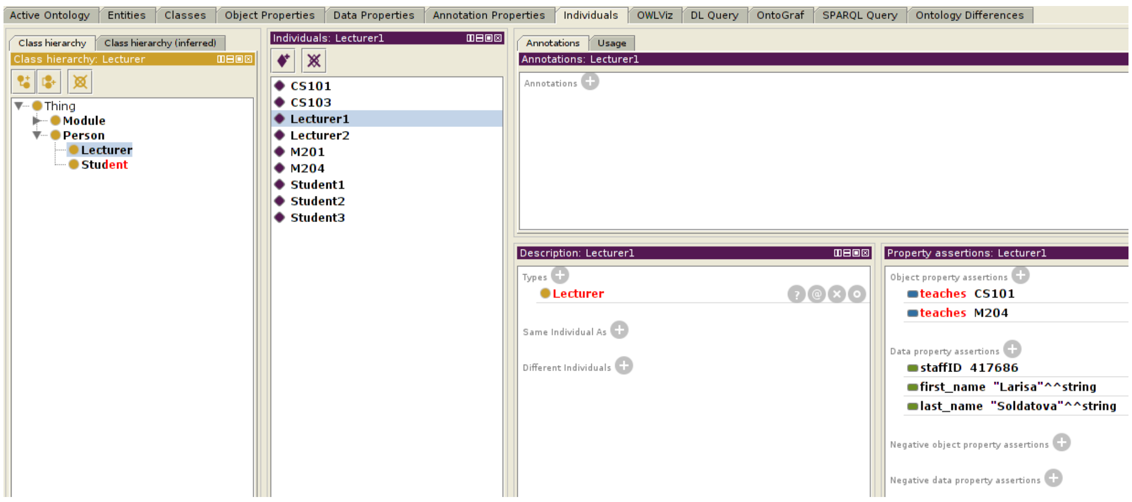
Figure 7: Protege - Data Properties in our Ontology

2.4 Adding some Individuals (Instances)

- Switch to the *individuals* tab
- Create new individuals *CS101* and *CS103*. Click on *Types* and choose *CSModule* for each of them
- Create new individuals *M201* and *M204*. Click on *Types* and choose *Math-Module* for each of them
- Create new individuals *Lecturer1*, *Lecturer2*. Click on *Types* and choose *Lecturer* for each of them
- Click on *Lecturer1* and:
  - add *Data property assertions*: *first_name* of type *string* and value *Larisa*
  - add another *Data property assertions*: *last_name* of type *string* and value *Soldatova*

  - add another *Data property assertions*: *staffID* of type *integer* and value *417686*

- Click on *Lecturer2* and add *Object property assertions*: *teaches CS103* and *teaches M201*
- Add Data properties for Lecture2 if you wish
- Create new individuals *Student1*, *Student2* and *Student3*. Click on *Types* and choose *Student* for each of them
- Click on *Student1* and add *Object property assertions*: *studies M204*, *studies M201* and *studies CS101*
- Click on *Student1* and:
  - add *Data property assertions*: *first_name* of type *string* and value *Josef*
  - add another *Data property assertions*: *last_name* of type *string* and value *Baker*
  - add another *Data property assertions*: *studentID* of type *integer* and value *226814*
- Click on *Student2* and add *Object property assertions*: *studies M204*
- Click on *Student3* and add *Object property assertions*: *studies M201* and *studies CS103*
- Add Data properties for Student2 and Student3 if you wish
- Your individuals should look like Figure 8



Class hierarchy: Lecturer
- Thing
  - Module
  - Person
    - Lecturer
    - Student

Individuals: Lecturer1
- CS101
- CS103
- Lecturer1
- Lecturer2
- M201
- M204
- Student1
- Student2
- Student3

Annotations: Lecturer1

Description: Lecturer1

Types
- Lecturer

Property assertions: Lecturer1

Object property assertions
- teaches CS101
- teaches M204

Data property assertions
- staffID 417686
- first_name "Larisa"^^string
- last_name "Soldatova"^^string

Figure 8: Protege - Individuals in our Ontology

2.5 Saving/Publishing the Ontology

- Save it in your local file as university.owl
- Make it available online
  (I am using `http://people.brunel.ac.uk/~csstnns/university.owl`)

3 Downloading & Running Apache's Jena Fuseki Server

- Download Apache's Jena Fuseki from:
  `http://jena.apache.org/download/index.cgi`
- Decompress the file and use your favourtite command line to *cd* into the resulting directory
- Execute the following command:

```
$ ./fuseki-server --update --mem /ds
```

- With this command, we have used settings for "ds" dataset in config.ttl, enabled updating the database with new data and create empty memory-based store
- If everyhing goes well, we should have our server running at:
  `http://localhost:3030/`

4 Apache's Jena Fuseki project

- Using your web browser, go to `http://localhost:3030/`
- Choose Control panel and then select "/ds" source
- In the File Upload section, select university.owl file from previous steps and upload it to server
- Now the database should be filled with some triples
- REMEMBER: OWL ontologies are collections of triples so querying them with SPARQL is similar to querying RDF triple stores!
- To view all the triples, go to SPARQL Query form and run:

```
SELECT * WHERE {?x ?y ?z}
```

- To view all the students who are studying M204 go back to SPARQL Query form and run:

```
PREFIX uni: <http://people.brunel.ac.uk/~csstnns/university.owl#>
  SELECT * WHERE { ?student uni:studies uni:M204}
```

- To view all the subclasses of class Person run:

```
PREFIX uni: <http://people.brunel.ac.uk/~csstnns/university.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?class
        WHERE { ?class rdfs:subClassOf uni:Person }
```

5 Another Ontology

Here we are going to create another simple ontology and place it on a different place (not at the same place as our previous ontology). We will see how we can query and retrieve data from the two ontologies at the same time. I am going to place this ontology on `http://www.meta-qsar.org/ontologies/sport.owl`

- Using the same steps from our university ontology, create a new ontology using protege
- Enter an Ontology IRI. I am using the following IRI:
  `http://www.meta-qsar.org/ontologies/sport.owl`
- Save it in your local file as sport.owl
- Create the classes shown in Figure 9

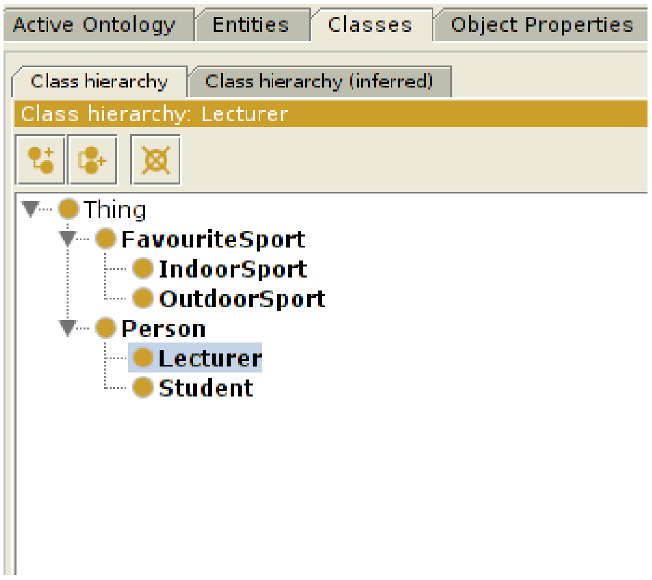


Figure 9: Protege - Classes in our Sport Ontology

- Add one object property *hasFavouriteSport* with domain *Person* and range *FavouriteSport*

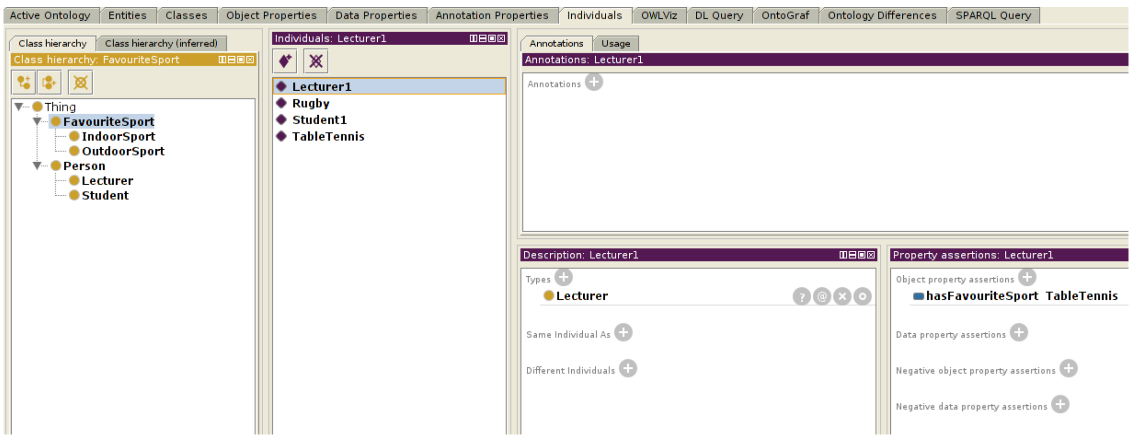


Figure 10: Protege - Individuals in our Sport Ontology

- Add the individuals shown in Figure 10
- Create new individuals *TableTennis* of type *IndoorSport* and *Rugby* of type *OutdoorSport*.
- Create new individuals *Lecturer1* of type *Lecturer* and *Student1* of type *Student*
- Click on *Lecturer1* and add *Object property assertions*: *hasFavouriteSport TableTennis*
- Click on *Student1* and add *Object property assertions*: *hasFavouriteSport Rugby*
- Save the ontology and publish it online
- If Fuseki server is still running, upload sport.owl (now we have sport.owl and university.owl in our dataset)
- To query both ontologies at the same time, for example to have a list of subclasses of class FavouriteSport and subclasses of class Person, run:

```
PREFIX sp: <http://www.meta-qsar.org/ontologies/sport.owl#>
PREFIX uni: <http://people.brunel.ac.uk/~csstnns/university.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?class
        WHERE {
                {?class rdfs:subClassOf sp:FavouriteSport}
                UNION
                {?class rdfs:subClassOf uni:Person}
              }
```

6 Querying the Remote Ontologies

I have made the university ontology available at `http://people.brunel.ac.uk/~csstnns/university.owl` and the sports ontology available at `http://www.meta-qsar.org/ontologies/sport.owl` in sections 2.5 and 5 respectively.

Observe that we have uploaded both ontologies to the Fuseki Server and queried them as we have seen in Section 5. That query was run against the local version (the one on Fuseki Server) and not against the *Remote* ontologies.

Now to query a remote RDF dataset in general, or our online available ontologies (bear in mind that owl ontologies are sets of triples), we can use SPARQL's *FROM* keyword as shown in the following query:

```
PREFIX sp: <http://www.meta-qsar.org/ontologies/sport.owl#>
PREFIX uni: <http://people.brunel.ac.uk/~csstnns/university.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?class
FROM <http://www.meta-qsar.org/ontologies/sport.owl>
FROM <http://people.brunel.ac.uk/~csstnns/university.owl>
WHERE {
        {?class rdfs:subClassOf sp:FavouriteSport}
         UNION
        {?class rdfs:subClassOf uni:Person}
}
```

I must admit I have been unsuccessful so far in running this query from Fuseki but it can be easily executed using the free program ARQ[1].

All you need to do is download ARQ, save the query in a text file (with .rq extension) and issue the command:

arq –query yourfile.rq

The result should be a list of subclasses of class FavouriteSport and subclasses of class Person!

P.S.
I have my own SPARQL Video tutorial series and the video that covers querying remote datasets using the FROM keyword can be watched from `https://www.youtube.com/watch?v=605iBiPRQ40&index=26&list=PLea0WJq13cnA6k4B6Tr1ljj2nleUl9dZt`

[1]`http://jena.sourceforge.net/ARQ`