Lab 4 Knowledge Graph and Ontology

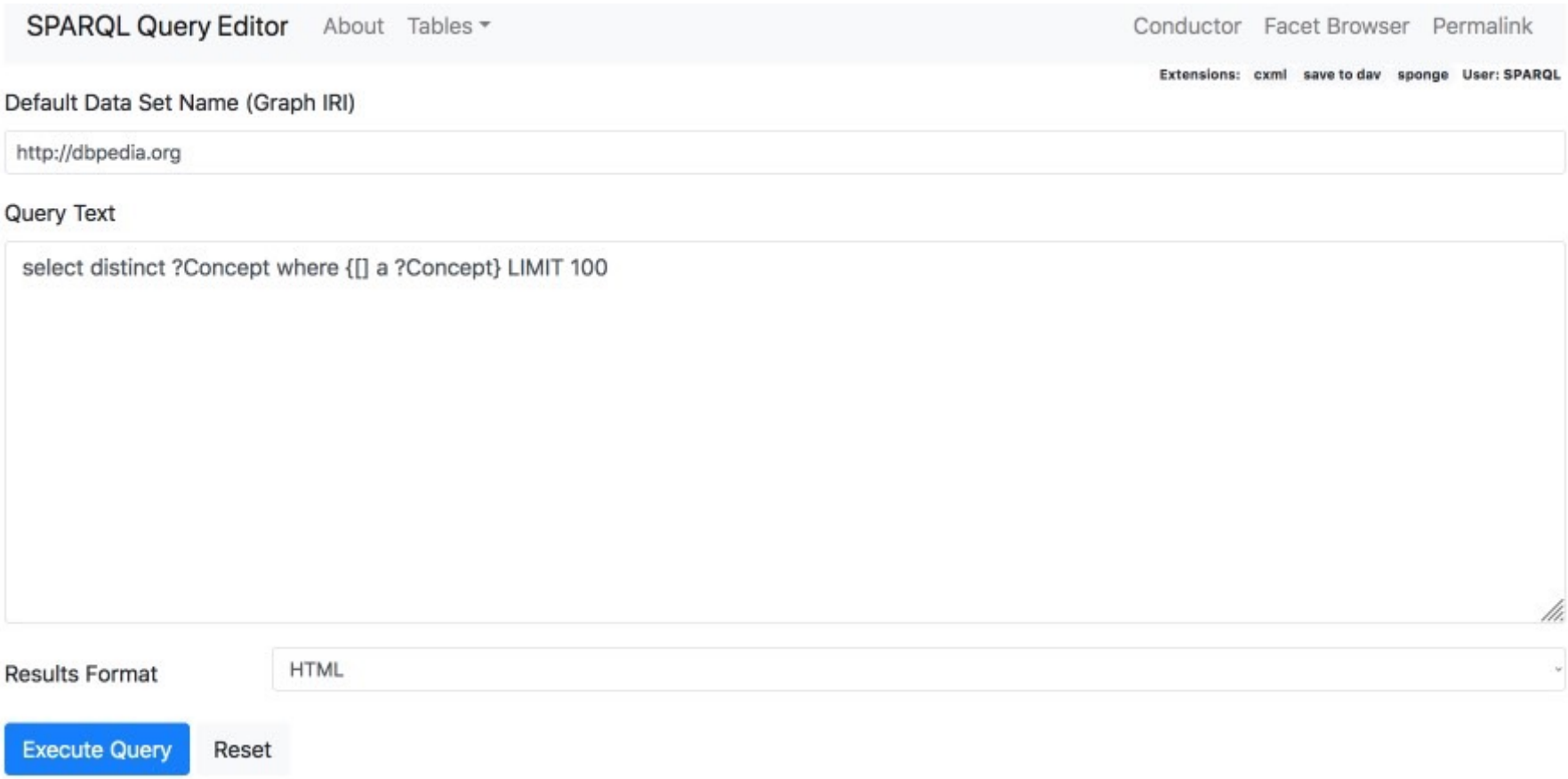
- The lab will enable you to
  - gain knowledge on RDF knowledge representation; RDF queries using SPARQL language; and Linked open data
  - develop an ontology using an ontology editing tool called Protégé, and design RDF data store with linked data technologies (e.g. RDF, OWL, ontology, etc).

Some slides on knowledge graph query are adapted from Daniel Heward-Mills: Running Basic SPARQL Queries Against DBpedia

DBPedia

- DBpedia is a community project that creates and provides public access to critical structured data for what's commonly referred to as the Linked Open Data Cloud.
- DBpedia provides a globally accessible Knowledge Graph derived from Wikipedia content. You can query this tremendous Knowledge Graph using the powerful SPARQL query language .
- Dbpedia can be accessed through its SPARQL endpoint.
  - http://dbpedia.org/sparql

DBPedia SPARQL Endpoint



RDF Data Query

- Knowledge takes the form of a collection of RDF Triples, which structures data using a (subject-predicate-object) object model
- The most basic query example can take the form of a SELECT Query where the triple-pattern in the Query Body comprises a subject, predicate, and object.
  - Query Solution projection size is limited to 10 records presented in an HTML Table in this lab.
- DBpedia can be easily queried both with and without a deep knowledge of the DBpedia ontology.

Query Example

- Go to: https://dbpedia.org/sparql and type the following query:

```
SELECT *
WHERE
   {
     ?s ?p ?o
    }LIMIT 10
```

SPARQL | HTML5 table

s	p	o
http://dbpedia.org/ontology/deathDate	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/birthDate	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/averageAnnualGeneration	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/foalDate	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/installedCapacity	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/birthYear	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/deathYear	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/diameter	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/displacement	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty
http://dbpedia.org/ontology/height	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#FunctionalProperty

Task 1 – SPARQL – Query 1

- Search for "an athlete with a literal value of Cristiano Ronaldo"

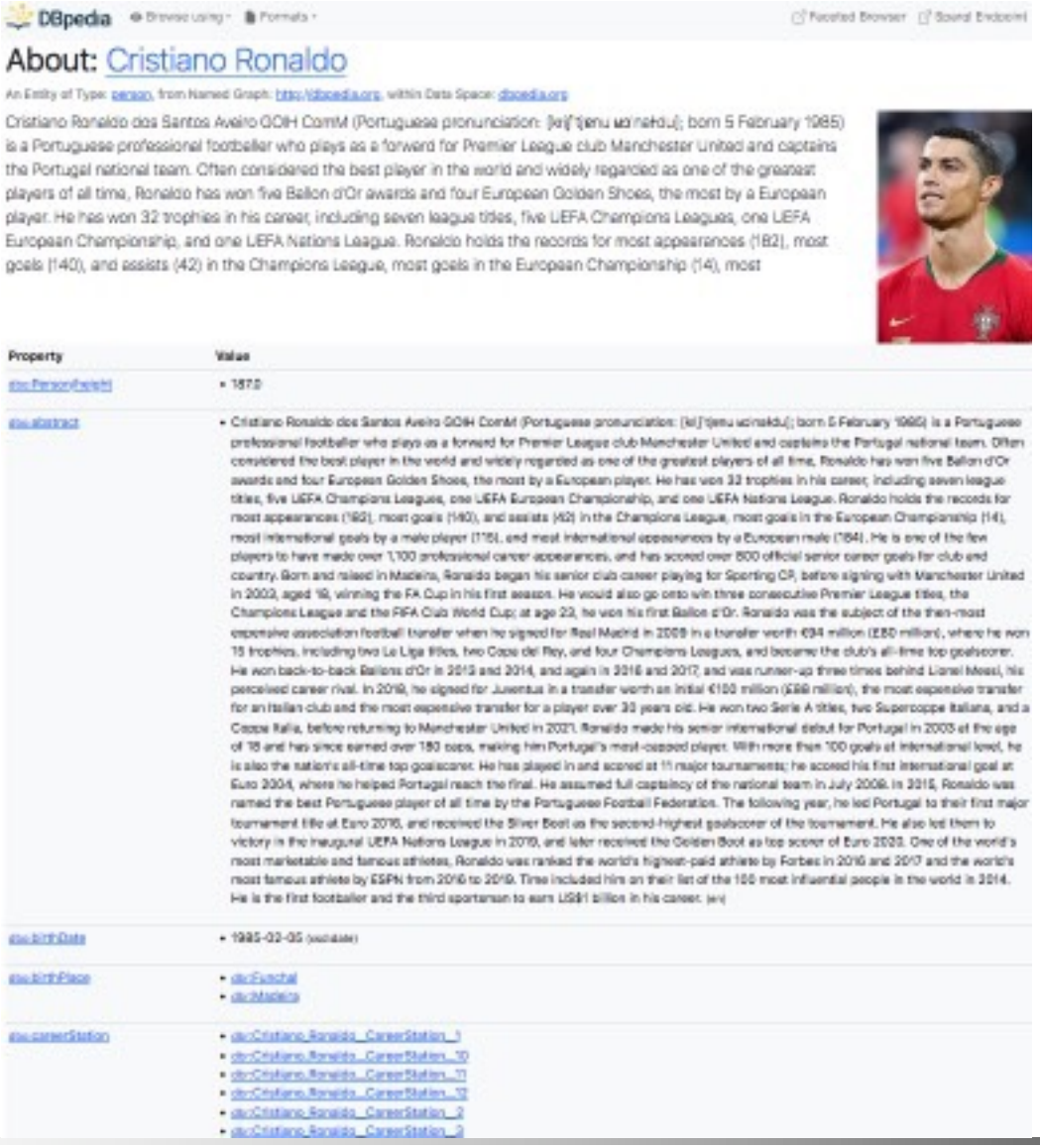
```
SELECT *
WHERE
    {
      ?athlete rdfs:label "Cristiano Ronaldo"@en
    }
```

- Record the output.

Dereferencing

- We can click on the resulting URI (lets try the first one) for Cristiano Ronaldo in the "athlete" column to view each relation associated with this specific URI.
- This process is known as dereferencing the DBpedia Identifier (an HTTP URI) that identifies the entity literally labeled as "Cristiano Ronaldo".
- Click on the URI and observe what you see.

Dereferencing – cont'd



Task 1 – SPARQL – Query 2

- Add in the dbo:birthPlace property.
- And execute the following query

```
SELECT *
WHERE
{
  ?athlete  rdfs:label      "Cristiano Ronaldo"@en ;
            dbo:birthPlace  ?place .
}
```

- Record the output.

Observation

- How can somebody have two birth places?
- Both results are correct, since Funchal is a city within the Autonomous Region of Madeira.
- Click on http://dbpedia.org/resource/Funchal for more information.
- We can narrow the place results to only include cities by scoping the query body to instances of the dbo:City class.

Task 1 – SPARQL – Query 3

- Execute the following query.

```
SELECT *
WHERE
{
  ?athlete  rdfs:label      "Cristiano Ronaldo"@en ;
            dbo:birthPlace  ?place .
  ?place    a               dbo:City ;
            rdfs:label      ?cityName .
}
```

- Record the output.

Task 1 – SPARQL – Query 4

- We can also limit the language tag to English via a FILTER on the ?cityName variable.

```
SELECT *
WHERE
{
  ?athlete  rdfs:label      "Cristiano Ronaldo"@en ;
            dbo:birthPlace  ?place .
?place    a               dbo:City ;
            rdfs:label      ?cityName .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

- Record the output.

Task 1 – SPARQL – Query 5

- Let's double-confirm that Funchal is in Madeira, by using the dbp:region property and its value.

```
SELECT *
WHERE
{
 ?athlete  rdfs:label     "Cristiano Ronaldo"@en ;
       dbo:birthPlace  ?place .
?place    a             dbo:City ;
          rdfs:label     ?cityName ;
          dbp:region     ?region .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

- Record the output.

Task 1 – SPARQL – Query 6

- We can also replace * in the SELECT List with a specific list of variables to be projected in the Query Solution.

```
SELECT
   ?athlete
   ?place
   ?region
WHERE
{
 ?athlete  rdfs:label     "Cristiano Ronaldo"@en ;
          dbo:birthPlace  ?place .
?place    a               dbo:City ;
          rdfs:label      ?cityName ;
          dbp:region      ?region .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

- Record the output.

Task 1 – SPARQL – Query 7

- Run the following query

```
SELECT *
WHERE
{
  ?athlete  rdfs:label      "Cristiano Ronaldo"@en ;
            dbo:birthPlace  ?place .
?place    a               yago:PhysicalEntity100001930 ;
            rdfs:label      ?cityName .
FILTER ( LANG ( ?cityName ) = 'en' )
}
```

- Record the output.

Developing Simple Ontology

- Go to http://webprotege.stanford.edu/ and familiarise yourself with the online ontology editing tool, Webprotege (need to register before use).
- Read the tutorials and guide:
  - https://protege.stanford.edu/publications/ontology_development/ontology101.pdf
  - https://protegewiki.stanford.edu/wiki/Main_Page

Preparation

- To make your work easier, I created a video tutorial. Hope that you will find it useful.
- Watch the video before asking ANY questions on using Webprotege.
  - https://meeting.tencent.com/v2/cloud-record/share?id=b9a14350-1da1-45de-80d0-94818b56be4f&from=3
  - password: dPfh

Preparation before Lab

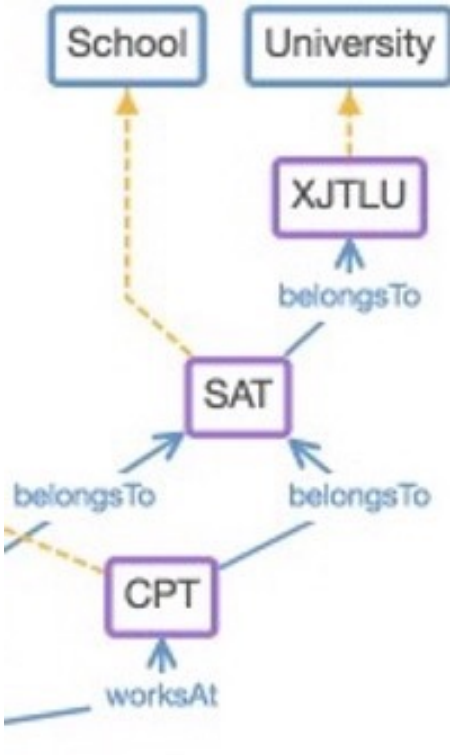
- After creating relevant classes and their relations, you need to create many individuals of those classes and link them together. This process is often referred to as ontology population (a basic step for creating linked data).
- Collection of these linked individuals forms a knowledge base, also known as knowledge graph. It can also be used to create the linked data, which is an extraordinarily large, distributed online data store.
- Lecture 10b covers a lot of what you need for this lab.

Task 2

- You are asked to create a very simple ontology for a university.
  - The following classes are suggested.
    - University
    - Schools
    - Departments
    - Professors
    - Students
  - Their relations need to be defined.
  - Individuals for the classes.

Task 2 – cont'd

- Generate a diagram shown most (or all) of the individuals, classes, and relations.
  - This can be generated using Webprotege. Click on the 'Individual' tab. You have many choices; choose an individual to generate an 'Entity Diagram' that contains most entities (e.g. individuals, classes, and relations).
  - Don't worry if the diagram does not include all the entities.
  - An example (a part of the diagram) shown at right side.
- Your diagram is used as the output for this task.



Task 2 – cont'd

- Ontology code: can be generated using Webprotege
  - Go to https://webprotege.stanford.edu/#projects/list, find the project, go to end of the line and click, select 'Download', choose 'RDF/XML' format, a file with 'owl' extension will be stored on your computer
  - This owl file is what you need to submit separately with your answer document.
  - An example can be found below.
- Print your code and attach it to your lab report.

```
    <!-- http://webprotege.stanford.edu/RCt0Yp7XExvsvG7LCvWaxze -->

    <owl:NamedIndividual rdf:about="http://webprotege.stanford.edu/
RCt0Yp7XExvsvG7LCvWaxze">
        <rdf:type rdf:resource="http://webprotege.stanford.edu/
R9uI2klx0RHqnSSFp8BlDQc"/>
        <webprotege:R7zuuVkF0ZD6msA3bcvr0kb rdf:resource="http://
webprotege.stanford.edu/R73iZt3H6LcwibCJh8kUbnO"/>
        <webprotege:RILs6ofEdOuTml1RNuUzAC rdf:resource="http://
webprotege.stanford.edu/RBQZfeBBTlony8gUqCRXm2J"/>
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">FP
at EEE, SAT, XJTLU</rdfs:comment>
        <rdfs:label xml:lang="en">LeoMessi</rdfs:label>
    </owl:NamedIndividual>
```