Semantic Data Management

Knowledge Graph Lab

Team members

Ali Arous

Dimitrios Tsesmelis

Section A

Before running the queries, we need to define the following prefixes

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://dbpedia.org/ontology/>
```

1. **Get the classes defined in the ontology.**

```
SELECT ?class
WHERE {
   ?class rdf:type owl:Class.
}
```

2. **Get the datatype properties defined in the ontology.**

```
SELECT ?property
WHERE {
   ?property a owl:DatatypeProperty
}
```

3. **Get the object properties defined in the ontology. What is the difference between datatype and object properties?**

```
SELECT ?property
WHERE {
   ?property rdf:type owl:ObjectProperty.
}
```

Datatype properties are relations between instances of classes, RDF literals and XML Schema datatypes. On the other hand, Object properties are relations between instances of two classes.

4. **Get the labels of all the properties (both datatype and object) defined in the ontology.**

As commented earlier, it is important to notice that the semantics of the knowledge graph are determined by different namespaces, which are defined by the "prefix" key-word at the beginning of the query. Thus, it is important to identify and understand the namespaces used in a *dataset.*

```
SELECT DISTINCT  ?label
WHERE {
{
        ?property a owl:DatatypeProperty;
        rdfs:label ?label
}
UNION
{
        ?property a owl:ObjectProperty;
        rdfs:label ?label
```

```
}
}
```

5. **Find the class representing an Actor in the dataset (using filters).**

```
SELECT ?actor_class
WHERE
{
  ?actor_class rdf:type owl:Class.
  FILTER EXISTS
  {
    ?actor_class rdfs:label "actor"@en
  }
}
```

6. **Find the super class for the class Actor.**

```
SELECT ?super_class
WHERE {
        :Actor rdfs:subClassOf ?super_class
}
```

7. **Find all the actors in the dataset.**

```
SELECT ?actor
WHERE
{
  ?actor rdf:type :Actor
}
```

8. **Get different classes that are defined as range of the properties that have the class Actor defined as their domain.**

```
SELECT DISTINCT ?class
WHERE {
        ?property rdfs:domain :Actor;
        rdfs:range ?class.
}
```

9. **Find the super property of the goldenRaspberryAward property.**

```
SELECT ?super_property
WHERE
{
  :goldenRaspberryAward rdfs:subPropertyOf ?super_property
}
```

10. **Return all the properties that have the class Actor as either their range or domain.**

```
SELECT ?property
WHERE{
```

```
        {?property rdfs:range :Actor}
UNION
        {?property rdfs:domain :Actor}
}
```

11. **Return all persons that are not actors.**

```
SELECT ?person
WHERE
{
   ?person rdf:type :Person.
   FILTER NOT EXISTS
   {
      ?person rdf:type :Actor
   }
}
```

12. **Return the path (in properties and classes) between the Actor and Person classes.**

```
SELECT ?a ?b ?c
WHERE {
{
        :Person ?a ?b.
        ?b ?c :Actor.
}
UNION
{
        :Actor ?a ?b.
        ?b ?c :Person.
}
}
```

Section B

1. **List the country, station type, latitude, and longitude details of each station.**
   Note: Limit the query to 25 results, and extract only the string values of the required object and not the whole IRIs.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>

SELECT DISTINCT ?station_name ?country_name ?station_type ?long ?lat
WHERE {
        ?station property:station ?station_name .
        ?station property:longitudeDegree ?long .
        ?station property:latitudeDegree ?lat .
        ?station property:type ?station_type .
   {
```

```
        ?station schema:inCity ?city .
        ?city schema:locatedIn ?country .
        ?country property:country ?country_name .
    } UNION
    {
        ?station schema:inCountry ?country .
        ?country property:country ?country_name .
    }
}
LIMIT 25
```

2. **List the 10 highest averages of C6H6 emission and the country and the year on which they were recorded.**
   Note: A sensor has a property (de_ned through the pre_x: <http://qweb.cs.aau.dk/airbase/property/>) stastisticShortName, and it can be Mean, Max, etc.

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?country_name ?year_as_number (avg(?c6h6) as ?avg_c6h6)
WHERE{
   ?observation schema:C6H6 ?c6h6 .
   ?observation schema:station ?station .
   ?observation schema:year ?year .
   ?year property:yearNum ?year_as_number .
   ?station schema:inCountry ?country .
   ?country property:country ?country_name .

   ?observation  schema:sensor ?sensor .
   ?sensor property:statisticShortName "Mean"^^xsd:string .
}
GROUP BY ?country_name ?year_as_number
ORDER BY DESC(avg(?c6h6))
LIMIT 10
```

3. **For each city and property type, give the yearly average emission for NO2, SO2, PB, and PM10.**

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?city_name ?pollutant (avg(?measured_val) as ?average_measured_value)
WHERE{
        ?observation ?p ?measured_val.
        ?observation schema:station ?station.
        ?station schema:inCity ?city.
        ?city property:city ?city_name.
```

```
        ?observation schema:year ?year .
        ?year property:yearNum ?year_as_number .
        ?observation schema:sensor ?sensor.
        ?sensor schema:measures ?comp .
        ?comp property:caption ?pollutant .
        FILTER (isLiteral(?measured_val)).
        FILTER regex(?pollutant, "NO2|SO2|PB|PM10", "i")
}
GROUP BY ?city_name ?pollutant ?year_as_number
ORDER BY ?city_name ?pollutant ?year_as_number
```

4. **Find the sensors that are placed in rural areas and detected the highest average rate of pollution (for c6h6) per year.** This query can be interesting in cases where we want to identify the exact location of rural regions where there is more pollution than usual (?sensor property:statisticShortName "**Max**"^^xsd:string).

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?sensor ?lat ?lon ?year_as_number (avg(?c6h6) as ?avg_c6h6)
WHERE{
   ?observation schema:C6H6 ?c6h6 .
   ?observation schema:station ?station .
   ?observation schema:year ?year .
   ?observation  schema:sensor ?sensor .
   ?sensor property:statisticShortName "Max"^^xsd:string .
   ?year property:yearNum ?year_as_number .
   ?station property:latitudeDegree ?lat .
   ?station property:longitudeDegree ?lon .
   ?station property:areaType ?areaType .
   FILTER (?areaType = "rural"^^xsd:string)
}
GROUP BY ?sensor ?lat ?lon ?year_as_number
ORDER BY DESC(avg(?c6h6))
LIMIT 50
```

5. **List the cities that have recorded the maximum measure of a certain pollutant for more than 30 years, along with the name of the pollutant and the number of recorded maximum times.**

```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?city_name ?pollutant (count(?year_as_number) as ?count)
WHERE{

        SELECT ?year_as_number ?city_name ?pollutant (max(?measured_val) as
        ?max_measured_value)
```

```
        WHERE{
                ?observation ?p ?measured_val.
                ?observation schema:station ?station.
                ?station schema:inCity ?city.
                ?city property:city ?city_name.
                ?observation schema:year ?year .
                ?year property:yearNum ?year_as_number .
                ?observation schema:sensor ?sensor.
                ?sensor schema:measures ?comp .
                ?comp property:caption ?pollutant .
                FILTER (isLiteral(?measured_val)).
        }
        GROUP BY ?year_as_number ?city_name ?pollutant

}
GROUP BY ?city_name ?pollutant
HAVING (count(?year_as_number)>30)
ORDER BY DESC(count(?year_as_number))
```

6. **For each pollutant list the countries by their total emission of it from the most polluting to the least along with the amount of that emission (alternative: list the 5 most polluting countries).**

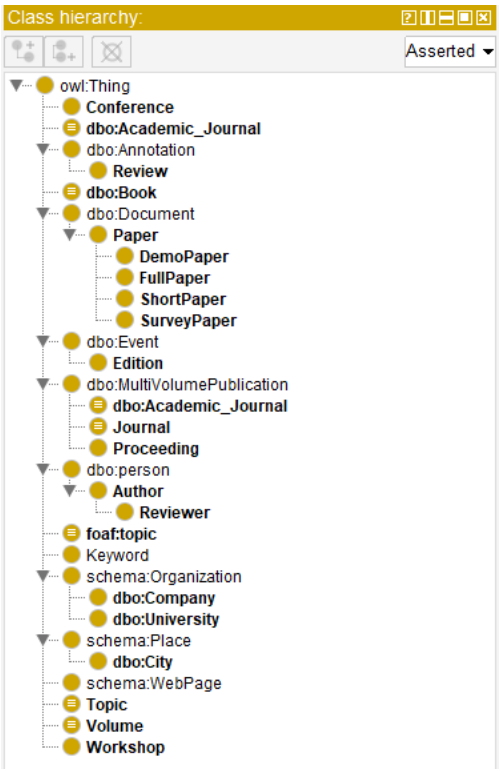
```
PREFIX schema: <http://qweb.cs.aau.dk/airbase/schema/>
PREFIX property: <http://qweb.cs.aau.dk/airbase/property/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?pollutant ?country_name (sum(?measured_val) as ?total_emission)
WHERE{
        ?observation ?p ?measured_val.
        ?observation schema:station ?station.
        ?station schema:inCity ?city.
        ?city property:city ?city_name.
        ?city schema:locatedIn ?country.
        ?country property:country ?country_name.
        ?observation schema:sensor ?sensor.
        ?sensor schema:measures ?comp .
        ?comp property:caption ?pollutant .
        FILTER (isLiteral(?measured_val)).
}
GROUP BY ?pollutant ?country_name
ORDER BY ?pollutant DESC(sum(?measured_val))
```

C1. TBOX definition

To construct the TBOX, we use the Protégé ontology editor (available for download at: https://protege.stanford.edu/) which is a free, open-source software written in Java, that facilitates the creation of TBOX triples though a user-friendly interface.

All definitions explained below, can be found in full statements in the file ***rdf scripts/publication.owl*** exported from Protégé and attached with this report.

The first part of our TBOX design in Protégé is the "Class hierarchy" design, shown below:



In this hierarchy, all classes are subclass of the owl:Thing at the root of the hierarchy, and some of the classes have child subclasses or associated equivalent classes. We use subclassing and association whenever possible in order to make the public research domain as complete as possible as well as to reuse concepts from existing ontologies (e.g. http://dbpedia.org/ontology/, http://xmlns.com/foaf/0.1/ and http://schema.org/). In this way , we make our ontology richer and well-connected to others.

The classes highlighted in our proposed class hierarchy are the following:

- **dbo:Document:** This concept is reused from DBpedia ontology (where dbo is used as a PREFIX of http://dbpedia.org/ontology/). We consider here that each research **Paper** is in the end a form of a document, so we subclass it from the generic Document concept. We also extend the Paper concept to distinguish among four types of research papers (namely: DemoPaper, FullPaper, ShortPaper and SurveryPaper). We show below the definition of both Document and Paper:

```
<owl:Class rdf:about="http://dbpedia.org/ontology/Document"/>
<owl:Class rdf:about="http://www.semanticweb.org/Paper">
    <rdfs:subClassOf
rdf:resource="http://dbpedia.org/ontology/Document"/>
</owl:Class>
```

- **dbo:Event:** This concept is reused form DBpedia ontology, as we look at the **Edition** of a given conference as a form of a scientific event being held at a certain place at a specific time, and so we subclass Edition from dbo:Event.
- **dbo:Person:** This concept is reused from DBpedia ontology. Here we consider each Author concept as a specialization of the Person concept. Likewise, we see any Reviewer as an Author who is playing a more specialized role when reviewing.
- **schema:Organization:** This concept is reused from (http://schema.org/) ontology, for unifying the subclasses dbo:Company and dbo:University (to which a certain author may affiliate) under the same type. We show below the definitions of Organization, Company and University:

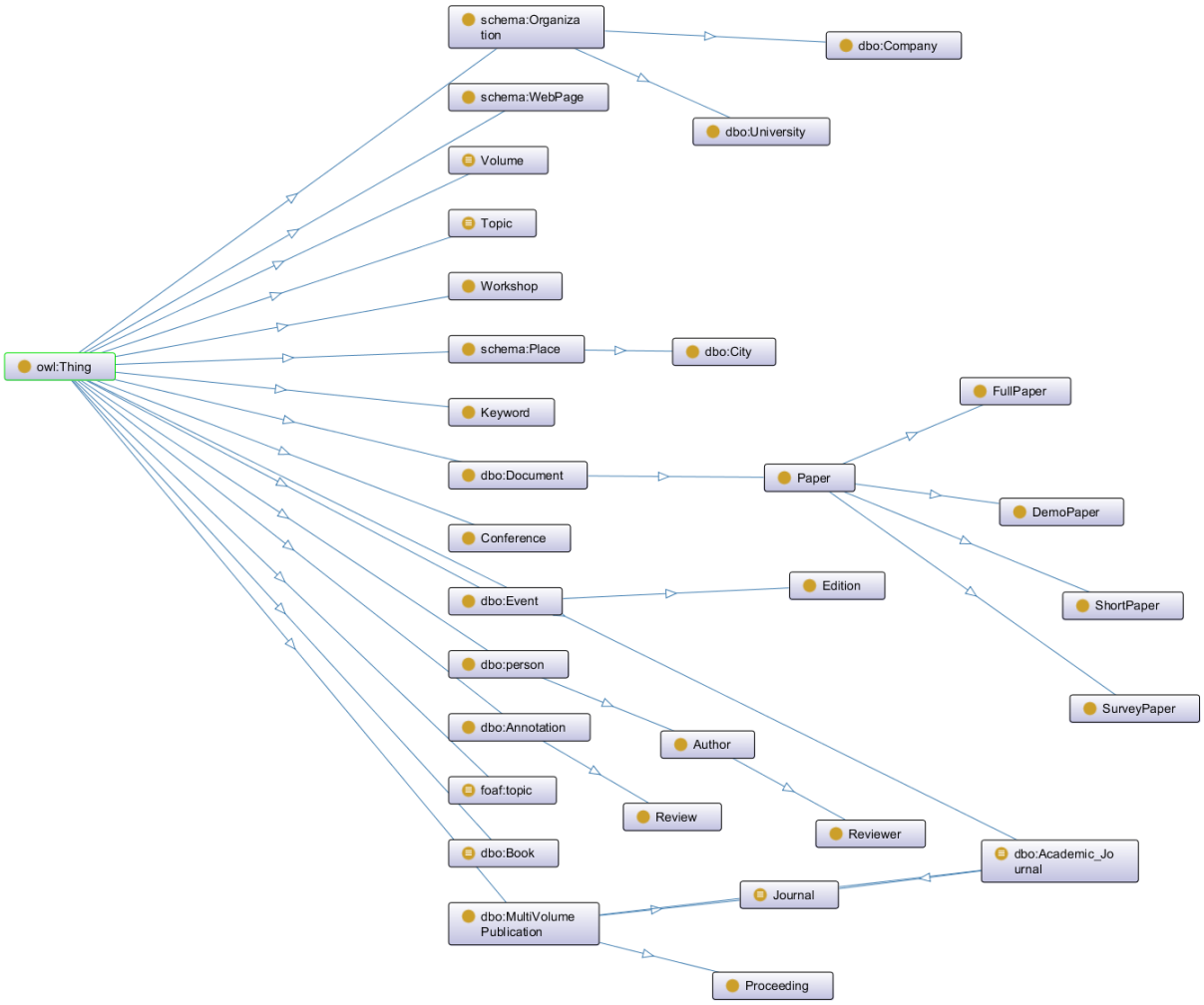
```
<owl:Class rdf:about="http://schema.org/Organization"/>

<owl:Class rdf:about="http://dbpedia.org/ontology/Company">
    <rdfs:subClassOf rdf:resource="http://schema.org/Organization"/>
</owl:Class>

<owl:Class rdf:about="http://dbpedia.org/ontology/University">
    <rdfs:subClassOf rdf:resource="http://schema.org/Organization"/>
</owl:Class>
```

- **dbo:MultiVolumePublication:** We reuse this concept from DBpedia ontology in order to have a base type for subclasses **Journal** and **Proceeding** seeing them as publications comprising multiple volumes.
- **dbo:Annotation:** We reuse this concept from DBpedia ontology as we think that it resembles a generalization of what a review concept may represent in the research domain. Thus, subclassing **Review** class from it.
- **dbo:Book** concept is reused as equivalent to our newly created **Volume** concept. In the same way, we reuse **dbo:Academic_Journal** as an equivalent concept to our newly created **Journal** one, and **foaf:topic** as an equivalent to our newly created **Topic** concept. We show below the definitions of foaf:topic and Topic:

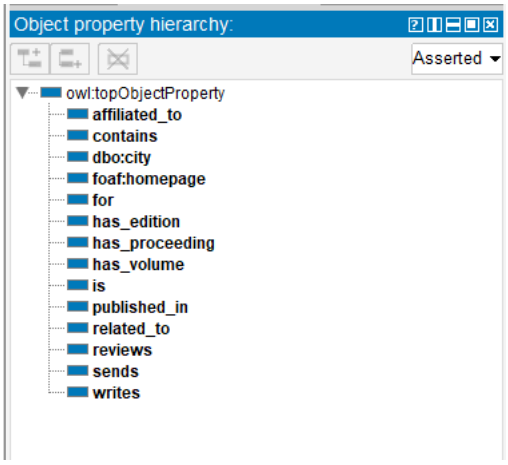
```
<owl:Class rdf:about="http://xmlns.com/foaf/0.1/topic"/>

<owl:Class rdf:about="http://www.semanticweb.org/Topic">
    <owl:equivalentClass
rdf:resource="http://xmlns.com/foaf/0.1/topic"/>
</owl:Class>
```

- **schema:Place:** This concept is reused from (http://schema.org/) ontology to generalize our newly created **City** concept that subclasses it. We believe that looking at the **City** as a class instead of a mere data property has a big advantage in making queries regarding places of publications more powerful, giving a room for mapping this concept to a hierarchy of places concepts like country, region or continent.
- **Conference, Workshop, and Keyword** are the only left newly created concepts in our ontology that do not map to corresponding concepts in external ontologies. However, due to the Object Properties linking these classes with other ones in our ontology, they are all indirectly linked with other concepts in external ontologies.
- **schema:WebPage** is the only reused concept from (http://schema.org/) ontology that does not map as a subclass, superclass or equivalent class to any other concept in our ontology, but is used as a range for **foaf:homepage** Object property that will be introduced later on in the report.

The visual representation of our TBOX created using Protégé, showing the subclassing relationship among ontology concepts:



The second part of our TBOX design in Protégé is "Object properties hierarchy" design, shown below:



In this hierarchy, each Object property is a subproperty of **owl:topObjectProperty,** while the list of them includes:

- **affiliated_to:** maps Author class in its domain to one of Organization subclasses in its range as follows:

```
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/affiliated_to">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/Author"/>
  <rdfs:range rdf:resource="http://dbpedia.org/ontology/Company"/>
  <rdfs:range rdf:resource="http://dbpedia.org/ontology/University"/>
</owl:ObjectProperty>
```

- **has_volume, has_edition, has_proceeding:** have similar semantics to each others, mapping each of Journal, Conference and Workshop in the domain to its composing elements in the range of the property, with one Journal having many volumes, and one Conference or Workshop having many Editions and only one Proceeding.

We show below an example of has_volume Object property definition:

```
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/has_volume">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/Journal"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/Volume"/>
</owl:ObjectProperty>
```

And here we show the multiplicity restriction on has_volume:

```
<owl:Class rdf:about="http://www.semanticweb.org/Journal">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.semanticweb.org/has_volume"/>
      <owl:someValuesFrom rdf:resource="http://www.semanticweb.org/Volume"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- **has_proceeding:** maps a Conference or a Workshop in the domain to one Proceeding in the range, as shown below:

```
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/has_proceeding">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/Conference"/>
  <rdfs:domain rdf:resource="http://www.semanticweb.org/Workshop"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/Proceeding"/>
</owl:ObjectProperty>
```

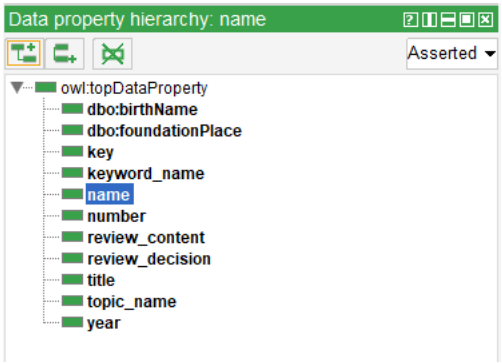
- **dbo:city:** This is a reused Object property from DBpedia ontology that maps each Edition of a Conference to the City it gets held in, as shows below:

```
<owl:ObjectProperty rdf:about="http://dbpedia.org/ontology/city">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/Edition"/>
  <rdfs:range rdf:resource="http://dbpedia.org/ontology/City"/>
</owl:ObjectProperty>
```

Other Object properties defined in our hierarchy are:

- **published_in:** maps a Paper in the domain to a Volume or an Edition in the range.
- **foaf:homepage:** maps one Journal, Conference or Workshop in its domain to one WebPage in its range.
- **writes, reviews:** maps Author and Reviewer respectively in the domain to Paper in the range.
- **sends:** maps Reviewer in the domain to Review in the range
- **related_to:** maps a Keyword in the domain to a Topic in the range
- **for:** maps a Review in the domain to a Paper in the range
- **contains:** maps a Paper in the domain to Keyword in the range

The third part of our TBOX design in Protégé is "Data properties hierarchy" design, shown below:



In this hierarchy, each Data property is a subproperty of **owl:topDataProperty,** while the list of them includes:

- **dbo:birthName :** This is a reused Data property from DBpedia ontology that specifies the birth name of an Author and has Author in its domain and xsd:string in the range (where xsd is used as a PREFIX of **http://www.w3.org/2001/XMLSchema#**), as shown below:

```
<owl:DatatypeProperty rdf:about="http://dbpedia.org/ontology/birthName">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/Author"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

- **dbo:foundationPlace:** This is also a reused Data property from DBpedia ontology that specifies the foundation place of dbo:Company or dbo:University in its domain and has xsd:string in the range, as shown below:

```
<owl:DatatypeProperty rdf:about="http://dbpedia.org/ontology/foundationPlace">
  <rdfs:domain rdf:resource="http://dbpedia.org/ontology/Company"/>
  <rdfs:domain rdf:resource="http://dbpedia.org/ontology/University"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

Other Data properties defined in our hierarchy are:

- **key:** specifies the key of Edition, Paper and Volume in the domain and has a xsd:string in the range.
- **name:** similar to key but instead of a key, it specifies a name for the three aforementioned classes.
- **year:** specifies the year of Edition, Paper and Volume in the domain and has a xsd:int in the range
- **keyword_name:** specifies the name of Keyword in the domain and has xsd:string in the range
- **number:** specifies the number of the Volume or Edition in the domain and has xsd:int in the range.
- **review_content** and **review_decision:** specify the content and the decision respectively of a Review in the domain, and they both map to xsd:string in the range.
- **title:** specifies the title of a Paper in the domain and has xsd:string in the range
- **topic_name:** specifies the name of a Topic in the domain and has xsd:string in the range

C.2 ABOX definition

To construct the ABOX, we use the same csv files that we used in the first lab of the course. The dataset can be found under the folder ***src/main/resources/input.*** For the transformation of the csv files into RDF, we use the Jena API. All the used transformation methods can be found inside the file ***src/main/java/Abox.java.***

To describe the method that we use to produce the RDF statements, we will use the following code as an example. This function creates the RDF statements for all the Authors in our ABOX.

```
public static void transformAuthor() throws IOException, URISyntaxException {

    Model model = ModelFactory.createDefaultModel();
    // read csv
    BufferedReader csvReader = new BufferedReader(
                               new FileReader(config.author_input));
    String row;
    while ((row = csvReader.readLine()) != null) {
        String[] row_data = row.split(",");
        String author_name = row_data[0];
        String affiliated_to = row_data[1].replace(" ","_");

        String authorUri = author_name.replace(" ","_");

        URI uri = null;
        try {
        URL url = new URL(config.RESOURCE_URL+authorUri);
        String nullFragment = null;
        uri = new URI(url.getProtocol(), url.getHost(),
                      url.getPath(), url.getQuery(), nullFragment);
        } catch (MalformedURLException e) {
            System.out.println("URL " + authorUri + " is a malformed URL");
        } catch (URISyntaxException e) {
            System.out.println("URI " + authorUri + " is a malformed URL");
        }

        Resource author =
model.createResource(config.RESOURCE_URL+uri.toString())
.addProperty(model.createProperty(config.DBO_URL+"birthName"),author_name)
.addProperty(model.createProperty(config.BASE_URL+"affiliated_to"),model.crea
teResource(config.RESOURCE_URL+affiliated_to));
    }
    csvReader.close();

    // write the mode to file
    model.write(new PrintStream(new BufferedOutputStream(
                new FileOutputStream(config.OUTPUT_FILE_PATH+"author.nt")),
true), "NT");
    }
```

As we can see in the above code, we read the csv line by line. In the beginning, we split each line to get the different columns (author_name and university). After that, we are forming the author's URI by using

the URI class. We took this decision because we faced several problems with the encoding of the names of the authors as they contained characters that were not accepted as valid URIs. Moreover, for every author, we create a resource under the **RESOURCE_URL** (http://www.semanticweb.org/resource/) which adds the property "birthName" to the author and a second resource which adds the property "affiliated_to". Finally, we are writing the generated RDF statements to the author.nt file.

Below, we can see a sample output of the above function for 2 different authors. The first triple relates the author *Jordan_Place* with the *Universidade_de_Fortaleza* and the second one adds to the same author the name "Jordan Place". We note here that for the property name, we reuse the birthName property of dbpedia ontology. Similarly for lines 3 and 4.

```
<http://www.semanticweb.org/resource/http://www.semanticweb.org/resource/Jordan_Place
> <http://www.semanticweb.org/affiliated_to>
<http://www.semanticweb.org/resource/Universidade_de_Fortaleza> .

<http://www.semanticweb.org/resource/http://www.semanticweb.org/resource/Jordan_Place
> <http://dbpedia.org/ontology/birthName> "Jordan Place" .

<http://www.semanticweb.org/resource/http://www.semanticweb.org/resource/Junji_Satake
> <http://www.semanticweb.org/affiliated_to>
<http://www.semanticweb.org/resource/Universidade_de_Fortaleza> .

<http://www.semanticweb.org/resource/http://www.semanticweb.org/resource/Junji_Satake
> <http://dbpedia.org/ontology/birthName> "Junji Satake" .
```

In total, we created 14 java functions and each one receives 1 .csv file as input and exports 1 .nt file as output. The names of the outputted files are: *paper.nt, author.nt, university.nt, link_author_paper.nt, keyword.nt, link_paper_keyword.nt, citations.nt, reviews.nt, journal.nt, volume_link_journal.nt, conference.nt, edition_link_conference.nt, link_paper_edition.nt and link_paper_volume.nt*. **The full source and output files can be found in the deliverable folder.**

C.3 Linking ABOX to TBOX

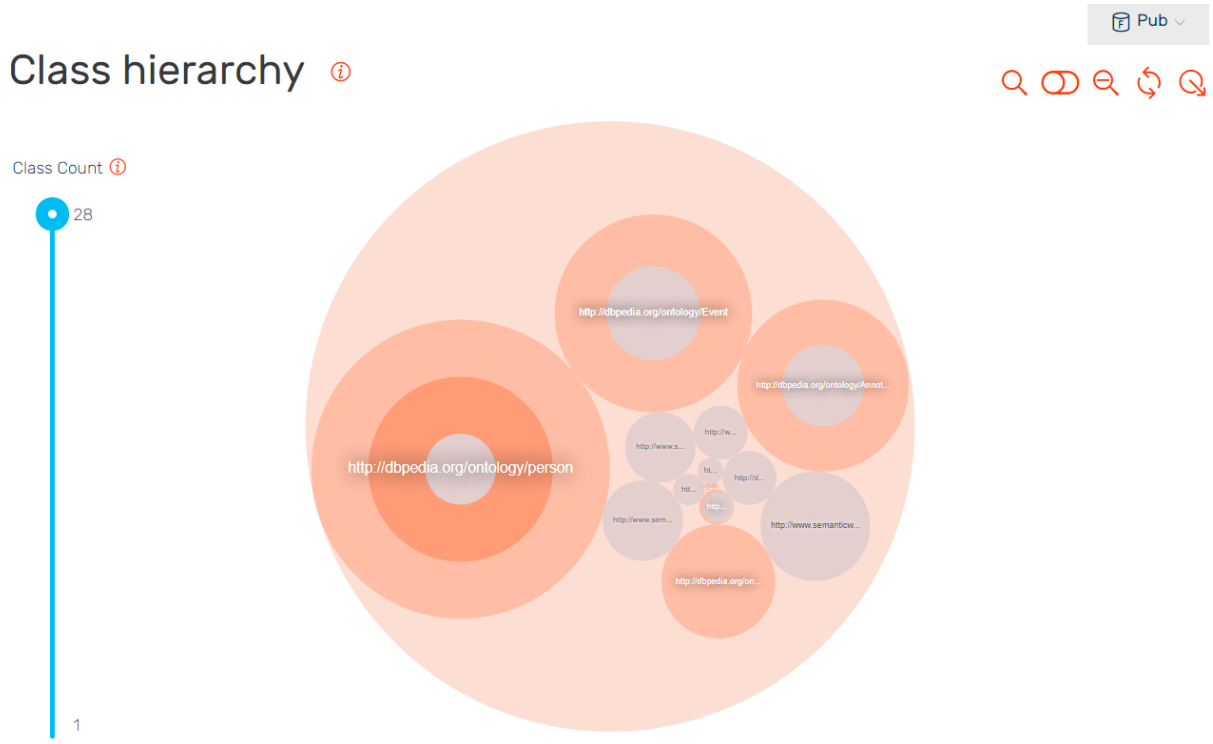
For the linking of the ABOX to the TBOX, we wrote the below SPARQL queries:

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

INSERT{?a rdf:type pub:Paper}
WHERE{
      ?a pub:key ?b .
      ?a pub:title ?c
};

INSERT{?a rdf:type pub:Author}
WHERE{
      ?a dbo:birthName ?b
};

INSERT{?a rdf:type pub:University}
WHERE{
      ?a pub:university_name ?b
};
INSERT{?a rdf:type pub:Keyword}
WHERE{
      ?a pub:keyword_name ?b
};
INSERT{?a rdf:type pub:Citation}
WHERE{
      ?a pub:citation_id ?b
};
```

```
INSERT{?a rdf:type pub:Review}
WHERE{
	?a pub:review_id ?b
};

INSERT{?a rdf:type pub:Reviewer}
WHERE{
	?a pub:sends ?b .
	?b rdf:type pub:Review
};

INSERT{?a rdf:type pub:Journal.
	?b rdf:type pub:Volume}
WHERE{
	?a pub:has_volume ?b
};

INSERT{?a rdf:type pub:Conference.
	?b rdf:type pub:Edition}
WHERE{
	?a pub:has_edition ?b
};
```

RDF Graph Statistics

For the linking of the ABOX to the TBOX, we imported the generated RDF data to GraphDB. The repository that we created had enabled the inference (RDFS-PLUS (Optimized)). In the first capture it is depicted the class hierarchy as it was exported from the exploration tool of GraphDB. After this image, we present some statistics about the obtained RDF graph.



Measure	Count
Number of classes	28
Number of Object properties	15
Number of Datatype properties	12
Number of instances	98663

Number of instances of the biggest classes:

Class	Count
Citations	40.011
Authors	26.046
Papers	11.841
Editions	10.000
Conferences	2.545

D Queries on top of the Ontology

1. **Find all the Authors (using the TBOX)**

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?author
WHERE {
        ?author rdf:type pub:Author
}
```

Find all the Authors (using only the ABOX)
Here we assume that in our ontology, only the authors have a property “writes” and that the object of such a property can only be a paper, as it has a property “key”.

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT distinct ?author
WHERE {
        ?author pub:writes ?paper .
        ?paper pub:key ?paper_key
}
```

2. **Find all the properties whose domain is Author (using the TBOX)**

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT distinct ?property
WHERE
{
        ?property rdfs:domain pub:Author.
}
```

Find all the properties whose domain is Author (using only the ABOX)
We make the same assumption as before.

```
PREFIX pub: <http://www.semanticweb.org/>
SELECT distinct ?property
WHERE
{
        ?author ?property ?object.
        ?author pub:writes ?paper .
        ?paper pub:key ?paper_key
```

```
}
```

3. **Find all the properties whose domain is either Conference or Journal (using the TBOX)**

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT distinct ?property
WHERE
{
        {?property rdfs:domain pub:Conference}
        UNION
        {?property rdfs:domain pub:Journal}
}
```

Find all the properties whose domain is either Conference or Journal (using only the ABOX)
Here we assume that only the conferences have a property named “has_edition” and only the journals have a property named “has_volume”.

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT distinct ?property
WHERE
{
        {
                ?conference pub:has_edition ?name .
                ?conference ?property ?object
        }
        UNION
        {
                ?journal pub:has_volume ?name .
                ?journal ?property ?object
        }
}
```

4. **Find all the things that Authors have created (either Reviews or Papers) (using the TBOX).**

```
PREFIX pub: <http://www.semanticweb.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT distinct ?thing
WHERE
{
        {
                ?subject pub:writes ?thing .
                ?subject rdf:type pub:Author
        }
        UNION
        {
                ?subject pub:sends ?thing .
                ?subject rdf:type pub:Reviewer
        }
}
```

Find all the things that Authors have created (either Reviews or Papers) (using only the ABOX).

```
SELECT distinct ?thing
WHERE
{
        {
                ?subject pub:writes ?thing .
                ?thing pub:key ?thing_key
        }
        UNION
        {
                ?subject pub:sends ?thing .
                ?thing pub:review_decision ?decision
        }
}
```